TURING 图灵原创



jQuery Mobile 开发指南

陆明 /著



人民邮电出版社
POSTS & TELECOM PRESS

数字版权声明



陆明　一位Web移动应用与Facebook营销应用开发专家和技术管理者，毕业于香港理工大学，在15年的从业过程中，经历了软件应用从PC到局域网，再到互联网和移动互联网的历次变革，曾经领导过多个互联网或移动互联网应用，从研发到上市。

此外，他还是一位徒步爱好者和摄影爱好者，喜欢在世界各地徒步旅游和拍摄风景。现居北京，从事数字营销产品的研发工作。



图书在版编目（CIP）数据

jQuery Mobile开发指南 / 陆明著. -- 北京 : 人民邮电出版社, 2014.2
（图灵原创）
ISBN 978-7-115-34371-0

Ⅰ. ①j… Ⅱ. ①陆… Ⅲ. ①移动电话机－应用程序－JAVA语言－程序设计 Ⅳ. ①TN929.53②TP312

中国版本图书馆CIP数据核字(2014)第013116号

内 容 提 要

jQuery Mobile 是一套基于 jQuery 的移动应用界面开发框架。本书将为你系统讲述使用该框架开发 Web 移动应用的方法，包括框架构成、页面、对话框、弹出页面、触控交互、按钮、工具栏、列表视图、表单，以及页面响应式布局设计、主题风格美化、事件响应与其他一些高级而实用的技巧。

本书既适合开发人员、测试人员和产品经理使用，也适合作为大中专院校相关专业师生的学习用书与培训教材。

◆ 著　　　陆　明
责任编辑　王军花
责任印制　焦志炜
◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　　　　印刷
◆ 开本：800×1000　1/16
印张：16
字数：378千字　　　2014年2月第1版
印数：1－3 000册　　2014年2月北京第1次印刷

定价：49.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京崇工商广字第 0021 号

前　　言

我从 2011 年开始关注 jQuery Mobile，当时它还是 1.0a2 版本，它的简单、高效和质量稳定很快就吸引了我。在学习 jQuery Mobile 几个或者十几个小时之后，一名 Web 工程师就能够快速开发出质量稳定而有趣的应用。难能可贵的是，基于 jQuery Mobile 所开发的应用能在绝大多数主流移动设备上顺畅运行。对于快速成长的创业公司来说，这极具吸引力，因为开发原生移动应用，需要支持多种不同的移动设备操作系统，这会给任何中小型软件创业团队带来不小的压力。事实上，jQuery Mobile 的潜力已经被越来越多的企业和组织所认可。在技术社区中，很多大公司正在支持基于 jQuery Mobile 的应用。在最近发布的 Visual Studio 2012 中，jQuery Mobile 被作为一种内置的应用解决方案包含其中。

坦率地说，写一本 jQuery Mobile 的书令我兴奋而忐忑。面对移动应用时代的到来，新技术呼唤着我，我兴奋不已，虽然这已经不是我第一次学习一种新技术。我的程序员生涯是从 20 世纪 90 年代在 Turbo C 下开发 DOS 程序开始的。上网冲浪也是从象牙塔中的 Telnet 和 Gopher 时代进入到 WWW 时代，再到今天的移动时代。在过去的十几年中，每次的变化都是一次蜕变。我很庆幸能目睹这些年来所发生的令人惊叹的变化。忐忑之中，希望我能跟上技术发展的脉搏，也能将这种新技术说清楚，讲明白。

本书的主要目标读者如下：

- Web 移动应用开发工程师；
- 移动应用创业者；
- 有兴趣学习移动应用开发的学生和工程师；
- 有兴趣研究 Web 移动应用的工程师。

本书会提供常用 jQuery Mobile 开发的范例程序，并结合实际应用场景探讨高级定制化的主题。这里我建议读者重点阅读基本的应用场景，因为这些场景用得最多。此外，书中的高级技巧在实际项目中也可能遇到，如果需要，可以查找书中相关内容。关于高级内容，读者只要能查到并会使用就够了。

本书共 13 章和 1 个附录，下面概要介绍各章的内容。

第 1 章概述了使用 jQuery Mobile 开发应用的优势和劣势，以及对于不同设备的支持情况。

第 2 章讲述了 jQuery Mobile 与 HTML5 之间的联系，介绍了开发一个基本的 Hello World 程序并通过 Web 服务器进行发布的过程。最后，我们还介绍了几种常用的 jQuery Mobile 开发工具。

第 3 章讲述了打开和关闭应用的页面和对话框，以及在不同页面和对话框之间进行切换的方法。

第 4 章讲述了页面的高级话题，包括通过预取和缓存改善页面访问速度、在页面之间传递参数、个性化定制加载消息以及支持 HTML5 的离线应用。在 jQuery Mobile 中，默认是无法实现命名锚记的，这里我们将讲述实现命名锚记的方法。

第 5 章讲述了弹出页面技术，这是在 jQuery Mobile 1.2.0 之后引入的新特性。这里将讲述基本的菜单、表单和对话框的弹出页面，此外还将介绍以弹出页面的形式呈现视频、图片和地图。

第 6 章讲述了虚拟鼠标事件和触控交互的实现方法。

第 7 章讲述了 jQuery Mobile 中的按钮。在 jQuery Mobile 中，除了表单元素中使用的按钮外，还会讲一些将超级链接美化为按钮的样式，比如实现页面切换的超级链接。在这一章中，我们还将讲述按钮的高级主题，包括自定义按钮图片以及按钮中的文字折行呈现等。

第 8 章讲述了页眉、页脚和导航工具栏，以及自定义图标的导航工具栏和自定义风格的工具栏。

第 9 章讲述了在 Web 移动应用开发中十几种常见的列表视图实现方法。此外，这一章还介绍了一些列表视图的触控操作、动态加载以及个性化界面设计的技巧。

第 10 章讲述了常用的表单和表单元素的使用方法，以及表单验证与文件上传的实现方法。

第 11 章讲述了页面布局的方法。基于这些内容，开发者可以在有限的用户界面中有效地组织和布局界面上的内容和功能。

第 12 章讲述了基于移动设备浏览器实现响应式设计的方法。从某种意义上说，第 12 章是第 11 章在响应式设计上的延伸，这里将重点探讨分栏技术、回流表格和字段切换表格的响应式设计以及滑动面板设计。

第 13 章讲述了对 Web 移动应用界面进行主题风格美化的方法，包括选择和设置 jQuery Mobile 内置色版，以及通过 ThemeRoller 进行风格与色版的定制化。最后，这一章将会讲述通过程序动态调整页面元素的主题设置方法。

附录讲述了进行 jQuery Mobile 开发时可能会用到的 JavaScript 单元测试自动化技术，这里的单元测试自动化采用 QUnit 实现，熟悉 jQuery 的开发者通常很快就能掌握这种测试自动化技术。

在这本书写作之初，我有幸得到谢子斌和杨海玲两位老师的指点和鼓励。在写作的过程中，图灵教育的王军花老师给予大量细致和耐心的指导和帮助。此外，我的家人也给了我很大的支持，使我每天都能安排完整的时间来学习和写作。大家的支持，是对我莫大的鼓励和鞭策。我在此对大家表示衷心的感谢！

jQuery Mobile 是一个发展非常迅速的技术，我希望通过一些知识与经验的分享，能为 jQuery Mobile 技术社区作出一些贡献。然而我深知自己能力有限，书中内容难免有所疏漏。如果你在阅读本书的过程中发现谬误或不足，请给予指正，非常感谢！

陆 明

2013 年 5 月

目　　录

第1章 jQuery Mobile概述



jQuery Mobile是一套基于jQuery的移动应用界面开发框架，以网页的形式呈现类似于移动应用的界面。当用户使用智能手机或平板电脑，通过浏览器访问基于jQuery Mobile开发的移动应用网站时，将获得与本机应用接近的用户体验。用户不需要在本机安装额外的应用程序，直接通过浏览器就可以打开这样的移动应用。

在这一章中，我们将会了解到：

- jQuery Mobile的基本概念；
- 应用jQuery Mobile的Web站点；
- Web移动应用和本机应用的差异，以及Web移动应用的优势；
- jQuery、jQuery Mobile和jQuery UI之间的关系；
- jQuery Mobile对于不同移动设备的支持情况；
- 其他流行的Web移动应用开发框架；
- jQuery Mobile许可协议；
- 受限的应用场景。

jQuery Mobile是基于jQuery JavaScript库和HTML5发展而成的移动应用用户界面系统。基于jQuery Mobile开发的移动应用，体积轻量，用户体验与界面风格统一，并兼容大量移动平台。在前端页面的呈现方面，jQuery Mobile实现了界面美化和对移动设备浏览器的兼容。在大多数开发场景下，jQuery Mobile应用开发不需要过多关心jQuery JavaScript库的实现方式，只需要在页面中引用合适的jQuery版本就可以了。

在jQuery Mobile出现之前，jQuery已经发展了很多年。jQuery于2006年发布了第一个版本，现在它已成为最受欢迎的JavaScript库。在全球访问量最高的10万个网站中，使用jQuery的网站高达52%。

另一个广泛流行的jQuery JavaScript用户开发框架是jQuery UI。jQuery Mobile与jQuery UI的定位不同：前者用于开发移动应用，后者常用于开发桌面Web应用。它们的共同之处在于它们都是基于jQuery JavaScript库实现的用户界面系统。

简单来说，jQuery Mobile应用就是一个Web移动应用。所不同的是，jQuery Mobile应用为移动设备在网络传输、页面呈现、用户行为交互等领域进行了特别的优化。在使用jQuery Mobile开发的移动应用中，业务逻辑通常主要通过Web服务器进行处理，而不是基于移动设备浏览器中

的jQuery Mobile页面来实现。在这样的应用中，HTML和CSS负责前端界面呈现，业务逻辑处理交予后台的Web服务器程序（可能是PHP、Python、JSP或者ASP.NET程序）。这些语言都可以用来开发jQuery Mobile的服务器端，这与其他网站开发是一样的。当讨论jQuery Mobile的时候，大多数情况下是关于如何使用jQuery Mobile开发适用于移动应用环境下的用户界面。

在当前的开源社区中，很多开源软件开发者基于jQuery Mobile贡献了许多有价值的插件。基于这些插件，移动应用开发者将可以快速、低成本而高质量地开发适用于多种移动设备的应用。

1.1　Web 移动应用还是本机应用

在进行移动应用技术方案选型的时候，首先要选择的是使用Web移动应用还是使用本机应用来实现开发需求。两种方案各有利弊，适用场景也有一些差异。

Web移动应用的优势在于，通过标准的HTML5语言以及浏览器支持能力，可以低成本地开发兼容性良好、跨移动平台的应用。在应用的部署过程中，可以不用依赖于电子市场或其他渠道商进行安装包版本检验和更新分发。和Web网站一样，Web移动应用更新或者重新部署到Web服务器之后，用户使用手机再打开这个网站，手机中的应用也就实现了同时更新。Web移动应用同样可以基于HTML5语言保存一定的用户本地数据，这样可以改善移动应用的运行速度。此外，Web移动应用不需要占用移动设备有限的存储空间。对于地理位置定位等应用，很多移动设备浏览器在支持HTML5语言的时候，也提供了相应的支持。这也为Web移动应用支持更多应用场景提供了便利。

Web移动应用在开发和运维过程中，会受到一定限制。假如移动网络速度比较慢或网络连接不够稳定，则Web移动应用的用户打开应用页面的速度会变慢，移动网络覆盖不到的地方则不能打开Web移动应用界面。此外，运行Web移动应用还可能产生网络流量费用。由于Web移动应用通过浏览器呈现界面并与用户交互，所以如果所应用的场景需要开发额外的手机底层应用，例如某种特定格式的视频播放器，则可能会受到限制。

传统的本机应用会在执行效率、使用过程成本和一些需要与硬件资源交互的环境下表现出明显的优势。不足之处在于，安装和部署成本高，推广会受到电子市场排行的影响、移动设备内存空间的限制，需要考虑到应用程序与移动设备的兼容性等。

相比之下，Web移动应用可以胜任大多数移动平台开发需求，包括内容订阅与分发、移动办公、远程监控、电子游戏和娱乐等。特别是在很多细分市场下，Web移动应用将非常具有优势，例如移动阅读。

1.2　移动平台兼容性

jQuery Mobile可以支持许多平台，包括台式机、智能手机、电子阅读器和平板电脑的操作系统等。对于不同的平台，支持程度有所区别，分为三种级别的官方支持，具体如下所示。

- A级：全部用户体验特效，包括基于Ajax的页面动画切换效果。
- B级：部分用户体验特效，一些Ajax特效和动画切换效果呈现将会受到影响。

- C级：基本的HTML功能特性，但没有增强的用户体验特效。

当前，大部分主流的移动应用平台均属于jQuery Mobile的A级范围内。

注意　Web移动应用使用的jQuery库版本可能影响jQuery Mobile的兼容性级别。例如，集成了jQuery 1.8.3库和jQuery Mobile 1.3.1的Web移动应用如果运行在Blackberry 5.0上，则只能获得C级呈现效果。如果使用jQuery 1.7.2库，则可以获得B级的呈现效果。

当然，在大多数设备中，jQuery Mobile应用是可以获得A级呈现效果的。

各种操作系统对 jQuery Mobile 1.3.1的支持情况如下所示。

- A级：
  - Apple iOS 3.2至Apple iOS 6.1；
  - Android 2.1至Android 2.3 ；
  - Android 3.2；
  - Android 4.0和Android 4.1；
  - Windows Phone 7.5至Windows Phone 7.8；
  - Blackberry 6至Blackberry 10；
  - Blackberry Playbook 1.0至Blackberry Playbook 2.0；
  - Palm WebOS 1.4至Palm WebOS 3.0；
  - Firebox Mobile 18；
  - Chrome for Android 18；
  - Skyfire 4.1；
  - Opera Mobile 11.5至Opera Mobile 12；
  - Meego 1.2；
  - Tizen；
  - Kindle 3和Fire；
  - Nook Color 1.4.1；
  - Chrome Desktop 16至Chrome Desktop 24；
  - Safari Desktop 5至Safari Desktop 6；
  - Firefox Desktop 10至Firefox Desktop 18；
  - Internet Explorer 8至Internet Explorer 10；
  - Opera Desktop 10至Opera Desktop 12；
  - 三星Bada 2.0；
  - UC浏览器。
- B级：
  - Blackberry 5.0；

  - Opera Mini 7；
  - Nokia Symbian ^3；
  - Internet Explorer 7（之前版本中IE 7是A级支持）。
- C级：
  - iOS 3.x或更早版本；
  - Blackberry 4.x；
  - Windows Mobile；
  - Internet Explorer 6或更早版本；
  - 其他更早版本的智能手机操作系统。

通过官方发布的清单可以发现，当前主流的移动平台，包括Apple iOS 3.2及其后续版本、Android 2.1及其后续版本、Windows Phone 7.5和Blackberry 6都对jQuery Mobile具有良好的兼容性。

对于很多中文界面的Web移动应用而言，建议还是尽可能对目标人群的主流移动平台进行界面样式和布局的测试，特别是中文字符集、字体、字号以及排版效果。

注意　对于不同版本的jQuery Mobile，其平台兼容性列表也会存在一定差异。开发者在选择某个jQuery Mobile版本之后，建议先检查移动平台与目标受众的移动设备是否能够良好兼容。例如，在早期版本中，Windows Phone 7是A级支持，而在jQuery Mobile 1.3.0中它不再作为A级支持。在jQuery Mobile 1.3.0之后，A级支持范围变为Windows Phone 7.5至Windows Phone 7.8。开发者需要根据应用场景选择合适的jQuery Mobile版本，否则早期的移动设备可能不能正常运行。

1.3　为何选择 jQuery Mobile

jQuery Mobile很有可能成为未来跨平台移动应用的主流用户界面系统之一。与其他跨平台移动应用相比，jQuery Mobile具有如下优势：

- 基于jQuery JavaScript框架，工程师资源和技术资源最为丰富；
- 框架轻量，适合移动应用场景；
- 良好的学习曲线和开发效率；
- 兼容所有主流移动设备平台；
- 支持HTML5语言；
- 对触屏和鼠标事件具有良好的支持；
- 面向移动设备的性能优化；
- 优化的移动应用界面风格；
- 支持领域广泛，包括手机、平板电脑和电子阅读器等；
- 框架开放，并允许扩展第三方插件；
- 开发了自定义主题、色版和字体风格的界面。

1.4 其他流行的 Web 移动应用开发框架

除了jQuery Mobile之外，还有其他一些基于HTML的Web移动应用框架。这些框架也已经独立发展较长时间，并成功应用在一些项目中。在这里我们将会简单介绍几种常见的移动Web应用开发技术：

- Sencha Touch；
- iUI；
- jQTouch。

1. Sencha Touch

Sencha Touch具有比较明显的优势，具体如下所示：

- 界面样式标准、美观，开发团队可以集中于应用功能实现，而对于美工方面的依赖较小；
- 快速的页面响应速度和美观的用户界面；
- 继承Ext JS下的良好学习曲线和开发效率；
- 用户界面风格统一；
- 基于Sencha Touch的Sencha Chart可以提供丰富的图表制作能力；
- 支持丰富的手势操作。

此外，Sencha Touch也具有一些局限性。Sencha Touch支持主流的移动平台，但相对jQuery Mobile而言数量略少。如果面对公众用户开展移动应用开发和部署，选择Sencha Touch就要谨慎一些。一些专注于特定行业领域应用的信息系统开发商在将移动应用迁移到Sencha Touch之后，取得了不错的效果。

如果希望了解Sencha Touch的更多信息，可以参见其官方网站：http://www.sencha.com/products/touch/。

2. iUI

iUI也是一种基于HTML的Web移动应用开发框架，旨在为iPhone和与iPhone相兼容的设备开发移动应用。该开发框架不但包含其他开发框架所具有的JavaScript框架和CSS，甚至包括常用的图标图片，保证了开发效率和界面一致性。

iUI的主要特性如下所示：

- 基于标准的HTML代码建立具有iPhone风格的导航菜单；
- 很少甚至不需要JavaScript就可以建立基于HTML的应用界面；
- 容易开发出具有iPhone风格的移动应用；
- 适应移动手机应用的界面设计要求。

如果大家想了解iUI的更多信息，可以参见其官方网站：http://www.iui-js.org/。

3. jQTouch

jQTouch也是一个基于jQuery的Web移动应用开发框架，大概在2009年诞生，它对Android、iPhone和基于WebKit核心的桌面浏览器支持较好。

只是jQTouch在纷繁的移动应用平台和移动浏览器框架的环境下，兼容性比较弱，所以并不建议使用jQTouch进行跨平台的移动应用开发，除非本身就是面对特定用户群体的。

如果大家想了解jQTouch的更多信息，可以参见其官方网站：http://www.jqtouch.com/。

1.5 jQuery Mobile 许可协议

jQuery Mobile同时支持GPL和MIT两种开源许可协议，这为开发者提供了更为灵活的法律框架，开发者可以针对不同的使用场景而选择不同的许可协议。这与jQuery是一致的。在Visual Studio 2012中，微软已经将jQuery Mobile作为一种移动应用解决方案集成在其中。

MIT是一种与GPL协议兼容，但比GPL、LGPL或者BSD更加宽松的协议。采用MIT协议的软件可以用于商业用途的开发和销售。MIT协议源自麻省理工学院（Massachusetts Institute of Technology，MIT），也被称为“X许可协议”（X License）或“X11许可协议”（X11 License）。

基于MIT协议，被授权人可以有权使用、复制、修改、合并、出版发行、分发、再授权及销售软件及软件的副本，也可以根据程序或者软件的需要而修改许可协议的内容，以满足商业应用的需要。需要在所有用到这个软件的副本中包含版权声明和许可声明，这是MIT协议对被授权人所需要承担的义务的要求。基于MIT许可协议中权利和义务的定义，从事商业化开发的软件企业可以安全地使用jQuery Mobile。

作为开源软件的开发和贡献者，你可以基于GNU GPL协议来进行二次开发，并基于GNU GPL协议再分发这样的开源软件。

1.6 受限的应用场景

jQuery Mobile是一种Web移动应用用户界面系统，大多运行在移动设备浏览器中，这也为jQuery Mobile的使用带来一些限制。例如，Web移动应用通常不能直接开发手机的手电筒功能，因为HTML不能直接操作手机里面的硬件设备。大多数情况下，这些场景都是与硬件集成相对紧密或者需要调用操作系统核心功能的应用。

很多移动应用不需要专门与硬件进行交互，例如Web邮件、博客或微博客户端、企业信息系统，这些场景通常是适合jQuery Mobile的。另外一些传统概念里必须与硬件设备集成才能运行起来的功能，在移动设备浏览器和HTML5的支持下，它们很多也能正常运行，例如地理定位功能。因为移动设备的基于GPS或者基于基站定位的功能已经集成到HTML5之中，所以事实上在这些场景下使用jQuery Mobile也没有太大的问题。

虽然在一些场景下，jQuery Mobile无法直接集成移动设备的核心功能，但一些专门针对HTML5的开发工具，例如PhoneGap，能够帮助开发者解决这些问题。基于PhoneGap所提供的一系列接口，开发者可以方便地集成移动设备中的地理定位、加速器、联系人、声音和振动等功能。应用程序开发完成之后，也可以通过PhoneGap封装成不同移动平台下的应用，包括Android、iOS、Windows Phone等，并部署到不同平台之上。

第 2 章 框架初探

2

使用jQuery Mobile开发一套Web移动应用页面与开发一套传统的Web站点页面，并没有很大差别。对于大部分工程师而言，熟悉jQuery Mobile的框架结构，了解基本的HTML5语言，通常很快就能开发出Web移动应用程序。因为jQuery Mobile已经基于移动界面的特点进行了大量规范化的封装。jQuery Mobile的优势就是它具有良好的学习曲线。

在这一章中，我们将会了解到：

- jQuery Mobile与HTML5的关系；
- jQuery Mobile框架的组成；
- 创建第一个jQuery Mobile应用“Hello World!”；
- 发布jQuery Mobile应用到Web服务器；
- 安装Android模拟器，并通过Android内置浏览器访问jQuery Moible应用；
- 常用的jQuery Mobile开发工具。

2.1 jQuery Mobile 与 HTML5

虽然在很多早期范例代码和网站中，jQuery Mobile可以通过定义`div`的方式与HTML 4页面集成，但是在最近发布的jQuery Mobile稳定版本中,需要使用 HTML5而不是HTML 4来开发Web移动应用。使用jQuery Mobile 1.0 alpha版本的开发者，需要特别小心这一点。

在jQuery Mobile中，经常用到的HTML5新特性包括：

- 简化而实用的语义标签；
- 增强的表单功能；
- 原生视频和音频支持；
- 绘图API；
- Web Sockets API；
- 离线Web移动应用；
- Web Storage；
- Web Worker多线程；
- 基于地理位置的 Geolocation API。

本节将介绍常用的HTML5技巧以及实际工作中与HTML 4相异的部分等基础知识，但不会重点探讨HTML5高级开发技巧。

2.1.1　HTML5 的演化

HTML 4.01是1999年发布的。当年，大量Web页面以静态页面呈现，一些需要支持动态呈现的功能往往通过JavaScript或者基于组件的技术来实现。之后随着互联网的发展，越来越多的动态内容被融入网站，传统HTML语言的不足逐渐显露出来。2006年，W3C与WHATWG（Web Hypertext Application Technology Working Group，网页超文本技术工作小组）开始合作开发新的HTML语言，在2008年他们发布了HTML5的工作草案。W3C计划在2014年发布HTML5标准化推荐标准。

与HTML 4.0.1和XHTML 1.0相比，HTML5草案增加了很多新的HTML标记，简化了一些早期的HTML标记，增强了功能，定义了异常处理和非法文档的细节，并使浏览器能够更好地处理语法错误。这使得开发者可以更加高效地开发Web应用，以更加清晰和简洁的代码实现曾经需要JavaScript才可以实现的功能。同时，一些传统的RIA（Rich Internet Application，富因特网应用）也被HTML所取代。

HTML5草案现在还不是W3C的推荐标准（W3C Recommendation），却已经被业界主流厂商广泛支持。在应用HTML5从事开发的过程中，需要特别留心目标受众人群的移动终端是否能够支持HTML5的新特性。建议开发者在使用某种HTML5新特性之前，根据目标受众人群的移动设备和平板电脑浏览器类型，并参照 http://mobilehtml5.org所发布的兼容性列表进行功能选择，最好在目标移动设备浏览器下进行兼容性测试。

2.1.2　HTML5 新特性

与HTML 4.01和XHTML 1.0相比，HTML5在一些HTML代码结构、界面样式定义和标记含义上都有一定的简化和重新定义。此外，HTML5还增强了一些新功能，如表2-1所示。

表2-1　HTML5新特性

新　特　性	含　　义
HTML标记	更新的HTML5标记
Canvas 2D	画布技术，通过脚本动态渲染位图图像
Web Messaging	跨文档消息通信。例如，向iFrame中的HTML发送消息
Web Sockets	基于TCP接口实现双向通信的技术
Drag and Drop	基于Web的拖曳功能
Microdata	实现语义网的技术，通过自定义Web页面词汇表扩展与实现语义信息
Audio Video	原生的音频与视频技术
Web Workers	基于JavaScript的多线程解决方案
Web Storage	将信息存储于浏览器本地。与Cookie不同的是，Web Storage存储的数据更多
HTML+RDFa	实现语义网的技术

此外，随着HTML5的发展，一些与HTML5相关的技术也被浏览器厂商所支持，例如 MathML以XML描述数学算法和逻辑，SVG以XML描述二维矢量图形等。

jQuery Mobile是一种面向浏览器的JavaScript界面实现方案。在一些场景下，适时地使用HTML5新特性将有助于增强和改善用户体验，增强Web移动应用的功能。例如，在实时监控应用中，通过Web Sockets实现移动设备与服务器之间的实时双向通信。在面对公众的内容发布系统上，例如新闻或者UGC社区，通过RDFa或者Microdata增强页面内容语义来优化搜索引擎。

2.1.3　jQuery Mobile应用中经常用到的新特性

使用jQuery Mobile 1.0 Alpha版本和Beta版本开发Web移动应用时，是基于HTML 4.01的，但是在jQuery Mobile 1.0发布之后，Web移动应用的开发已经转为基于HTML5。特别是jQuery Mobile 1.3.0所增加的响应式设计等新特性，需要基于HTML5才能运行。

在基于jQuery Mobile的Web移动应用中，经常用到的HTML5新特性如表2-2所示。

表2-2　jQuery Mobile应用中常用的HTML5新特性

新　特　性	应用领域
DOM选择器	大多数jQuery Mobile选项、属性和事件处理中将会用到
增强的表单功能	jQuery Mobile表单
Media Queries	面对高分辨率屏幕的用户界面设计与图片呈现。 屏幕方向发生变化之后的页面布局调整。 响应式设计，jQuery Mobile 1.3.0之后开始支持。
Session Storage	在多页面模板环境下实现参数传递
离线Web应用	移动应用运行的网络环境通常并不稳定，可能在2G与3G移动网络之间切换或者在移动网络与Wi-Fi之间切换甚至断网。在网络不可用的时候，通过离线Web应用这个特性，可以改善用户体验

此外，还有一些HTML5新特性也会根据业务场景需要而应用于Web移动应用中，例如通过画布特性渲染图像，或者通过Geolocation实现位置定位服务等。

2.2　下载 jQuery Mobile

基于jQuery Mobile开发的移动应用通常包含以下部分：

- jQuery JavaScript文件；
- jQuery Mobile JavaScript文件；
- jQuery Mobile层叠样式表；
- jQuery Mobile图形资源库。

随着jQuery Mobile新版本的发布，其执行性能和移动设备的兼容性越来越好。开发者可以从

http://jquerymobile.com/download/下载最新版本的jQuery Mobile库。此外，建议开发者定期检查jQuery Mobile的版本更新情况。

如果将jQuery Mobile代码托管在自己的服务器上，建议下载ZIP格式的压缩文件，这个压缩文件包括jQuery Mobile库、层叠样式表以及与之配套的图片资源，但并没有包括jQuery JavaScript库。若想下载 jQuery JavaScript库，可以访问http://www.jquery.com/。

开发者可以按照下面的方法在HTML页面中集成jQuery Mobile:

- 将jQuery Mobile托管在网站自己的服务器；
- 在页面中访问http://code.jquery.com站点中托管的代码资源。

对于上面的两种方法，我们建议使用第一种方法，这是因为国内用户连接国外互联网资源的速度都比较慢。如果将jQuery Mobile相关文件部署在国内的服务器，则可以大大加快移动设备的访问和呈现速度。

小经验　有条件的话，建议将Web移动应用的CSS、JavaScript和图片等静态文件通过CDN分发。在选择部署Web移动应用的IDC机房时，这个机房的线路一定要有很好的网络品质，还要能保证与各个移动运营商的网络连接通畅。测试从移动运营商网络下访问Web移动应用时，尽量选择尽可能多的城市。有时候，即便在同一个运营商网络中，一个城市能够访问到Web移动应用，换一个城市后却不可以访问了。这对于支持2G网络环境下的Web移动应用尤其重要。

如果希望直接使用jQuery所托管的代码资源，则只需要在HTML页面head标记中加入下面的代码即可：

```
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.css" />
<script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
<script src="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.js"></script>
```

小经验　在jQuery Mobile官方网站的下载页面或ZIP格式的压缩文件中，你会看到xxx.min.js或者xxx.min.css文件，比如jquery-1.7.1.min.js、jquery.mobile-1.3.1.min.js、jquery.mobile-1.3.1.min.css，同时也可以看到不包含“min”文字的文件。

两种文件的区别在于包含“min”的文件对JavaScript或者CSS进行了压缩，删除了不需要的回车、空格键等内容，这样文件体积更小。如果只是调用而不是调试，建议部署min文件，这样将有助于减少移动网络流量，提升移动设备打开HTML页面的速度。

2.3　第一个程序

如同所有软件开发教程一样，我们的第一个范例也是Hello World!。

2.3.1 开发前的准备

Web移动应用的核心是通过HTML页面实现的，因此调试包括jQuery Mobile在内的Web移动应用与调试传统网页程序的方式基本上一致。建议开发者首先在支持HTML5的桌面浏览器上调试通过，然后再进行移动设备的兼容性测试。这里推荐使用安装了Firebug的Firefox浏览器。如果移动设备浏览器的兼容性在jQuery Mobile的官方支持列表中属于A级，那么兼容性测试通常都会顺利通过。

在开发Web移动应用时，最好能在基于桌面浏览器的测试之外，还能够针对用户特点安装移动设备模拟器。在模拟器环境中，模拟通过移动设备访问移动应用，并由此检查Web移动应用设计中的缺陷和待改进的特性。开发者也可以将Selenium测试系统与移动设备浏览器集成，以实现跨平台的验收测试和回归测试自动化。

在从事jQuery Mobile开发和测试的时候，建议能够部署某个Web服务器用以发布和测试这个Web移动应用。Web服务器可以是IIS（Internet Information Server，Internet信息服务）、Apache HTTP服务器或者其他支持HTTP协议的Web服务器。

2.3.2 Hello World!

一个简单的jQuery Mobile应用程序与通常的HTML页面差别不大，通常包含这样几个部分：基于HTML5的代码框架、jQuery JavaScript库和jQuery Mobile JavaScript库。

这里我们介绍的第一个jQuery Mobile应用为Hello World！，其代码如代码清单2-1所示。

代码清单2-1 Hello World!

```
<!DOCTYPE html>
  <html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="js/jquery-1.7.1.min.js"></script>
    <script src="js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
        <h1>jQuery Mobile跨平台移动应用</h1>
      </header>
      <div class="content" data-role="content">
        <p><a href="Section02/HelloWorld.html"> Hello World!</a></p>
      </div>
      <footer data-role="footer">
        <h1>页脚</h1>
      </footer>
    </section>
  </body>
</html>
```

在上面的框架中，主要涉及下面的内容。

- <!DOCTYPE html>：它是HTML5新的标签定义，表示一个HTML5文档的开始。在任何使用jQuery Mobile实现的页面中，必须以<!DOCTYPE html>开始。
- <html>、<head>和<body>：这些标签继承自HTML 4标准，表示HTML文档的各个部分。
  - <html>标签表示HTML文档的开始，而</html>则表示文档的结束。
  - <head>和</head>之间用于放置HTML文档的头部信息和引用，包括JavaScript引用和CSS引用。jQuery JavaScript文件和jQuery Mobile JavaScript文件就需要标记在这部分。
  - <body>和</body>则包含能够被呈现给用户的部分。
- <meta charset="utf-8">：表示页面内容的字符集，用于多语言显示。
- <meta name="viewport" content="width=device-width, initial-scale=1">：表示适合移动设备屏幕。如果忽略设置viewport，将导致移动应用显示内容过小而难以阅读。

在jQuery Mobile开发中，单页模板表示一个网页包含一个页面，这与传统的网页开发一致。在jQuery Mobile的多页模板开发中，一个网页可以包含多个页面，并通过Ajax方式实现页面跳转，具体请参见3.1节。

在不同的移动设备和分辨率下，除了设置viewport之外，还可以通过Media Queries技术实现在不同屏幕方向和分辨率下载入不同背景图片或CSS样式设计的功能。关于viewport和Media Queries的具体介绍，请参见11.1节。

在<body>标签中，我们定义了这样几部分内容。

- <section data-role="page">：用于设置data-role属性为page，表示所在section容器中的内容为一个页面。
- <header>标签所包含的部分是页面中的页眉，而<footer>标签所包含的部分则是页面中的页脚。页眉和页脚除了用于必要的信息显示之外，还可以通过添加导航按钮实现导航功能。有关页眉、页脚的使用方法，请参见第8章。
- <div class="content" data-role="content">：表示其内容为移动应用界面的正文部分。

使用Google Chrome浏览器运行该程序，得到的结果如图2-1所示。



图2-1　Chrome下所呈现的Hello World!

在Android 4.0.3模拟器中运行该程序，得到的界面效果如图2-2所示。

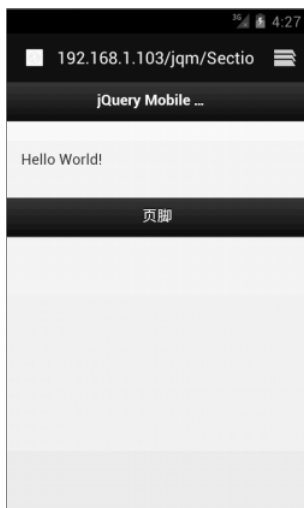


图2-2 Android 4.0.3下所呈现的Hello World!

至此，我们完成了第一个用jQuery Mobile开发的Web移动应用程序。

2.4 发布应用

一般情况下，移动设备通过HTTP或者HTTPS的方式访问Web应用。因此，Web应用只需要发布到HTTP服务器并能够被移动设备访问到，就算完成了发布。这里我们简单介绍一下基于IIS实现Web应用发布的过程。

IIS是运行在Windows之上的一种常见HTTP服务器，可以安装在Windows 7、Windows Server 2008等操作系统上。很多互联网服务商也提供基于IIS的虚拟机或虚拟主机，采用这种方案来搭建和部署自己的Web移动应用的成本较低。

在实际生产环境中，很多网站或Web移动应用除了采用IIS之外，也会采用Apache、Nginx、Lighttpd或者直接使用JBoss或者Tomcat搭建。这些都能提供HTTP服务，只是环境不同，各有利弊。

2.4.1 安装IIS

下面以Windows 7环境为例，简要介绍IIS的安装和部署过程。

首先检查Windows 7中是否已经安装了IIS。打开“控制面板”，选择“程序和功能”，选择“打开或关闭Windows功能”。此时如果发现“Internet信息服务”为选中状态，则表示已经安装了IIS，如图2-3所示。否则，就需要安装IIS。



图2-3　安装IIS

点击“Internet信息服务”之后，默认会选择大部分最常用的IIS功能。当然，开发者也可以进一步检查IIS安装模块和功能，核实自己所需要的功能是否都被安装上。在检查是否安装IIS的时候，建议检查“常见HTTP功能”项，将主要功能都选择并安装。通常，需要至少包括“HTTP错误”、“静态内容”和“默认文档”3个功能。否则，实现一些功能或者解决常用问题时将会遇到困难。

安装完IIS之后，在“控制面板”下的“管理工具”中可以找到“Internet信息服务（IIS）管理器”。打开IIS管理器，界面如图2-4所示。



图2-4　安装完成后的IIS管理器

此时启动服务器，打开地址http://127.0.0.1/，通常可以看到如图2-5所示的页面。



图2-5 IIS 7欢迎页面

2.4.2 通过IIS发布Web移动应用

在安装完IIS之后，就可以使用移动设备浏览器访问网站内容。将Web应用发布到IIS有两种方法：一种是以Web站点方式发布，另一种是以虚拟目录或应用程序的方式发布。

如果移动应用直接以Web站点的形式发布，那么当使用者打开浏览器，输入这个站点的地址时，就可以直接打开这个应用。

很多情况下，Web移动应用会和其他桌面Web应用发布在同一个站点下，但各自使用不同的目录。此时，就可以将Web移动应用部署在这个Web站点下的一个虚拟目录或者应用程序中。当网络用户浏览虚拟目录或IIS站点的应用程序时，输入的URL地址看起来是这个域名下的一个目录。

本书的演示内容都是静态内容，所以在发布到IIS的时候，我们使用的是虚拟目录的发布方式。对于现实应用中的Web移动应用而言，因为大多是动态网站，所以一般都部署为独立Web站点或应用程序，而不是虚拟目录。

这里我们以将jQuery Mobile移动应用部署在一个名为jqm的虚拟目录为例，演示在IIS环境中发布Web移动应用的方法。其他部署在网站根目录或应用程序的方法与部署虚拟目录的方法大致相当。

首先启动“控制面板”，选择“管理工具”，接着选择“Internet信息服务（IIS）管理器”，然

后选择所要发布的网站，这里我们会选择默认的网站Default Web Site。然后在选中的网站上单击鼠标右键，选择“添加虚拟目录”，如图2-6所示。

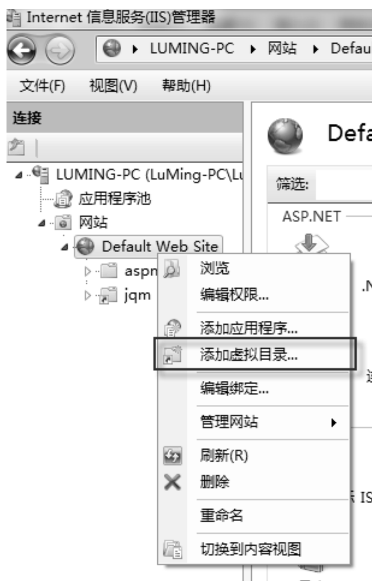


图2-6　添加虚拟目录

此时将打开“添加虚拟目录”对话框，如图2-7所示，在其中输入希望用户访问的路径名和所指向的页面地址，就可以实现虚拟路径的发布。在后续的代码范例中，我们将采用jqm作为虚拟路径发布Web移动应用，因此在之后的演示代码中将可以看到包含有jqm字样的URL地址。

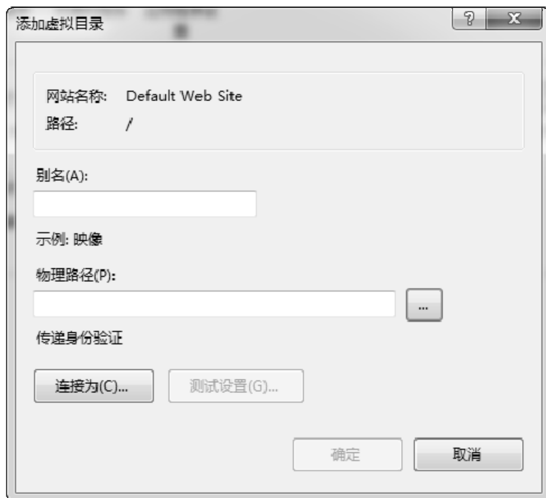


图2-7 “添加虚拟目录”对话框

在本书所有范例的运行环境中，我们都是通过部署在IIS的虚拟目录下来发布的。对于不同的开发环境，开发者也可以选择IIS之外的应用服务器部署移动应用，例如Apache服务器或者Nginx服务器等。

2.5 移动设备模拟器

基于jQuery Mobile开发的移动应用可以运行在大多数桌面操作系统上，开发过程的大部分时间可以基于桌面浏览器来进行，这将有助于提升开发效率。在正式发布应用之前，我们建议使用移动设备模拟器或实际的手机或平板电脑来执行用户验收测试，以确保最终的用户体验与预期的一样。移动设备模拟器是一种可以运行在PC上的虚拟化技术。基于模拟器，用户可以在Windows、Mac或者Linux等操作系统中模拟某种移动操作系统，并在这个操作系统中安装和调试所开发的应用。Android模拟器的运行界面如图2-8所示。



图2-8 Android模拟器的运行界面

2.5.1 安装Android模拟器

如果要安装Android模拟器，首先需要检查计算机中是否安装了JDK，如果没有安装JDK（Java SE Development Kit），则需要先安装JDK。Java JDK的下载地址为http://www.oracle.com/technetwork/java/javase/downloads/index.html。

接着下载Android SDK，其下载地址为http://developer.android.com/sdk/index.html。

> **注意**　截至笔者完稿时，Android SDK还不能运行在Windows 8环境中，所支持的Windows操作系统的最高版本为Windows 7。

按照提示下载完Android SDK后，就可以安装和配置Android模拟器环境了。在Windows操作系统中，进入“所有程序”，展开Android SDK Tools菜单，点击Android SDK Manager菜单项，进入Android SDK Manager管理工具，如图2-9所示。开发者可以通过这个工具管理和下载所需要的不同Android版本的模拟器安装包。

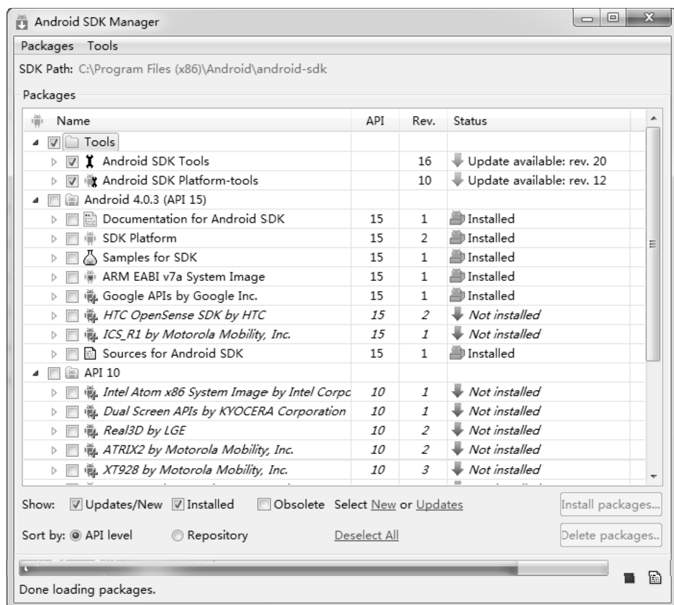


图2-9　Android SDK Manager

在安装各个包的时候，会提示许可协议接受界面，如图2-10所示。在接受之后，安装过程会继续下去。

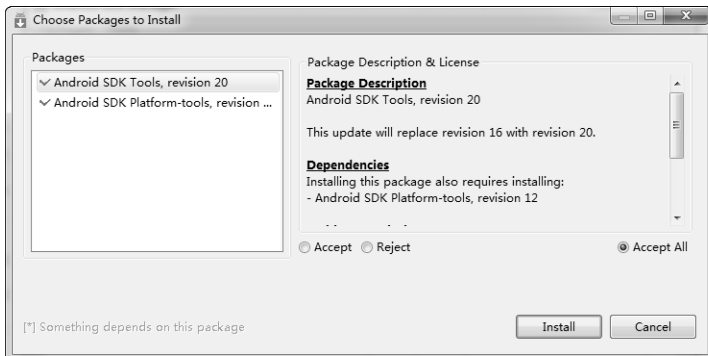


图2-10　许可协议接受界面

本书所使用的模拟器环境为Android 4.0.3，开发者可以根据业务场景需要选择安装合适的模拟器环境。

2.5.2　创建Android模拟器

在通过Android SDK Manager下载和安装模拟器之后，可以通过Android Virtual Device Manager创建所需的Android虚拟设备。开发者可以为每一个独立的Android运行场景单独创建Android虚拟设备，例如平板电脑、手机或者其他嵌入式设备，以便在独立的运行环境之间进行开发、部署和测试。

在用户体验和很多细节表现上，Android虚拟设备并不如实际移动设备操作方便，特别是触控操作方面。而从开发、调试、集成与测试的成本和效率来看，首先在移动设备模拟器上进行开发，然后再迁移到实际手机或平板电脑进行测试，将有助于提升开发生产率，快速稳定产品的质量，加快产品投放市场的速度。

在Android SDK中，可以通过Android虚拟设备管理器来创建、编辑或者删除不同的虚拟设备。在Windows操作系统中，点击“所有程序”菜单，展开Android SDK Tools菜单，点击AVD Manager便可以进入Android Virtual Device Manager界面，如图2-11所示。

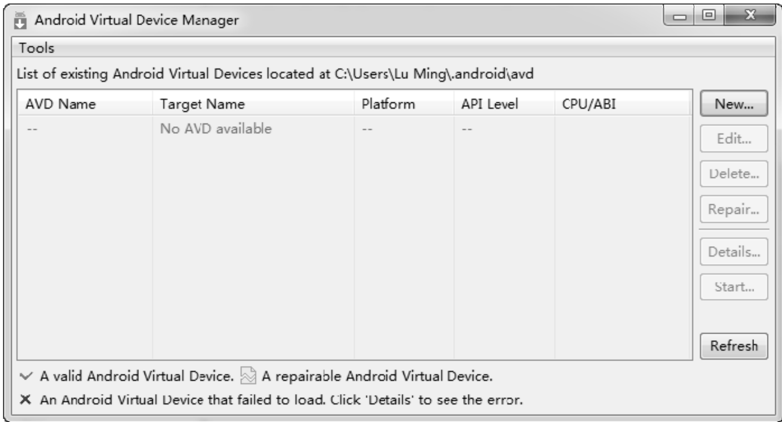


图2-11　Android Virtual Device Manager界面

点击New按钮，可以在Create new Android Virtual Device(AVD)界面中创建所需的Android虚拟设备，如图2-12所示。我们将这个虚拟设备命名为jQueryMobile，并选择目标运行平台为Android 4.0.3 – API Level 15。

小技巧　Snapshot选项表示对虚拟设备建立快照，建议开发者选中这个选项。这样虚拟设备将能够通过建立快照的方式帮助开发者保留上次开发与操作的内容，并能在下次启动虚拟设备的时候，直接进入上次关闭前的状态。这可以省去Android系统每次重新启动的时间。对于开发jQuery Mobile应用而言，这是一个提升开发效率的技巧。

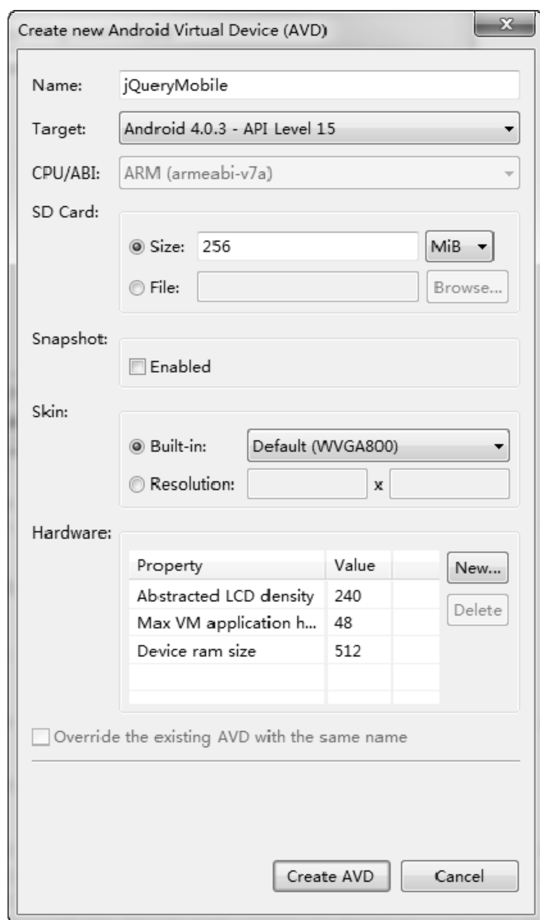


图2-12　Create new Android Virtual Device对话框

在点击Create AVD按钮之后，会打开一个确认对话框，如果确认对话框和所要创建的Android虚拟设备一致，则点击确认。在Android虚拟设备管理器列表中，选择要启动的虚拟设备，例如刚刚创建的名为jQueryMobile的虚拟设备，并点击管理器右侧的Start按钮，则可以启动虚拟设备。

在创建AVD的时候，如果选择了Snapshot选项，则可以直接基于上次关机时的快照将虚拟设备运行起来。例如，上次关闭虚拟设备的时候，浏览器停留在某个页面，则这次打开将直接恢复到上次关闭的位置。这就是快照功能。图2-13所示为Launch Options对话框，其中Launch from snapshot表示从上一次关闭的快照中恢复和运行虚拟设备，Save to snapshot表示在退出模拟器的时候，Android虚拟设备的快照将会自动保存。

启动虚拟设备的时候，可以根据目标设备的尺寸设置屏幕尺寸和分辨率，例如三星Galaxy Note I9200为5.3寸屏，Galaxy S3 I9308为4.8寸屏，I9100为4.3寸屏。

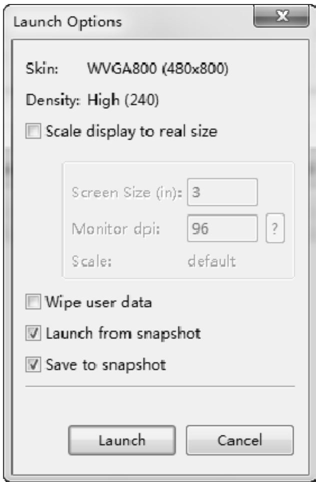


图2-13　Launch Options对话框

2.5.3　使用Android模拟器

在使用Android虚拟设备的时候，可以直观地看到界面所呈现的内容，如图2-14所示，开发者可以基于键盘或者模拟器界面中的按键进行界面操作。



图2-14　水平显示与垂直显示的Android模拟器

虽然大部分操作都可以通过鼠标点击Android模拟器中的键盘和按钮来实现，但是使用键盘快捷方式可以加快用户界面的操作速度，特别是一些模拟器操作几乎只能基于键盘快捷方式来实现，例如全屏显示、调整水平显示或者垂直显示。表2-3列出了常用的一些Android模拟器快捷方式。

表2-3　Android模拟器快捷方式

功　　能	快捷方式
返回Home屏幕	Home
菜单键Menu	F2或Page Up
Star	Shift+F2或Page Down
返回	Esc
拨号	F3
挂断电话结束通话	F4
查找	F5
电源按键	F7 长时间按下，进入关机菜单
水平方向与垂直方向切换	Ctrl+F11
水平方向与垂直方向切换，与上一条方向相反	Ctrl+F12
启动或关闭网络	F8
启动Code Profiling	F9 启动模拟器，需要`-traces`选项
全屏模式	Alt+Enter
轨迹球模式	F6
临时进入轨迹球模式	Delete

小经验　此外，还有一些其他的Android模拟器快捷方式，但是它们与使用jQuery Mobile开发Web移动应用关联不是很大，在此就不一一列举，例如增加或减少音量、启动摄像头等。如果你希望获得更多信息，可参见http://developer.android.com/tools/help/emulator.html。

2.6　jQuery Mobile 开发工具

jQuery Mobile开发与通常的HTML5页面开发没有特别大的差别，其核心是一套JavaScript与Themes的定义。只要支持HTML5开发的主流开发工具，一般都可以用于jQuery Mobile开发。

下面推荐几种开发工具：

- Notepad++；
- UltraEdit；
- 安装有Aptana插件的Eclipse；
- Adobe Dreamweaver；
- Visual Studio 2010或Visual Studio 2012；
- MyEclipse。

这样的工具还有很多，开发者可以根据自己的开发环境与喜好进行选择。

在调试HTML5代码的时候，推荐使用Firefox浏览器并集成Firebug工具、Google Chrome浏览器的“开发者工具”或者IE 10/IE 11所带的“F12开发人员工具”。

开发者可以使用Firebug检查Web页面的执行状况、调试、记录日志、优化性能、监控网络状况、调试和开发页面CSS Layout层等。这将有助于提升开发Web应用的质量和效率。图2-15是使用Firebug观察jQuery Mobile的Hello World!程序的界面截屏。

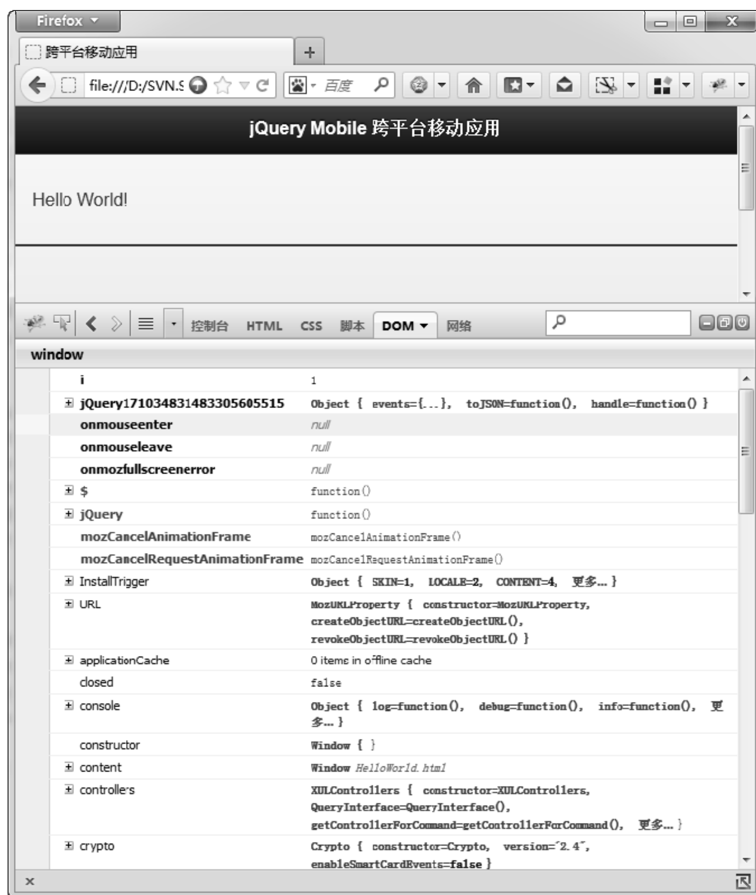


图2-15 使用Firebug观察Hello World!程序

开发者首先可以下载和安装Firefox（其下载地址为：http://www.firefox.com.cn/），然后通过Firefox下载和安装Firebug插件（其下载地址为：http://getfirebug.com/）。

如果使用Google Chrome的开发者工具或者IE的开发人员工具，开发者可以通过快捷键F12来打开这个工具的界面进行开发与调试。

第3章 页面与对话框

了解页面与对话框开发技术是学习jQuery Mobile移动开发的第一步。页面和对话框就好像一个“容器”，移动应用的页面元素都要放在这个容器中。

在这一章，我们将会了解到：

- 单页模板与多页模板；
- 页面标题；
- 链接到内部页面或外部页面；
- 建立和关闭对话框；
- 切换方式。

3.1 单页模板与多页模板

单页模板和多页模板是使用jQuery Mobile开发页面和对话框时经常用到的技术。基于jQuery Mobile开发Web移动应用时，如果一个网页文件只包含一个页面，那使用的就是单页模板技术；而一个网页文件中包含多个页面，并能够通过链接在这个网页文件所包含的各个页面间或不同网页文件的页面间跳转，这使用的就是多页模板技术。

形象地说，多页模板类似于网页开发中的选项卡。为了节省用户打开页面选项卡的时间，开发者会在HTML页面中将选项卡的内容从服务器下载并缓存到浏览器中。当用户打开某个选项卡时，浏览器将不再需要重新到服务器获得页面内容，只需将选项卡中隐藏的内容显示出来即可。

台式机具有稳定而高速的网络带宽。相比之下，在Web移动应用中，移动设备使用2G网络、3G网络或者Wi-Fi连接到Web服务器，网络通信环境非常复杂。移动设备的网络传输速率受到连接到移动基站的速率、基站自身的带宽品质、一个基站所覆盖的移动设备的数量，以及移动设备在运行过程中从一个移动基站切换到另一个移动基站时重新建立连接的影响，所表现的网络品质很不稳定，甚至会出现网络切换中的频繁下线和上线。我们乘坐地铁时打开网页的速度慢于在马路上等公交车时，其原因就在于此。所以，Web移动应用就需要能够针对多种复杂的网络环境进行专门的优化设计。

在jQuery Mobile中，多页模板在提升页面的加载速度、改善用户体验方面更进一步。用户在下载第一个页面的时候，同时也下载了相关的其他页面。当用户打开这样的页面时，不再需要重

新从服务器下载新的页面，而只需要将“隐藏”在背后的页面呈现出来就好了。这是多页模板应用的典型场景。

注意 在基于多页模板的设计中，一个网页文件中不同页面之间的切换不需要再重新建立连接和下载，因为所有内容都一次性下载到移动设备中了。

一次下载多页模板页面的所有内容可以减少网络传输，但是多页模板的设计存在一个局限，那就是它不适用于HTML页面中DOM元素的尺寸过大或过于复杂的场景。如果HTML页面中DOM元素的尺寸较大或者复杂，移动设备浏览器的处理速度可能会变慢。这就需要用到jQuery Mobile的高级技术——页面预存和缓存技术。页面预存和缓存技术能够帮助开发者在网络连接速度和DOM解析效率之间获得平衡，相关内容可参见第4章。

第2章中所接触到的Hello World!是一个典型的单页模板页面。

多页模板基于单页模板添加新页面，并能够通过超级链接跳转到新的页面。基于单页模板实现多页模板时，大致需要两个步骤，具体如下所示。

(1) 在单页模板页面之后添加一个新的页面部分。通常会使用`<section>`容器来作为各个页面的容器，在`<section>`中设置`data-role`属性为`page`即可实现一个页面。这与单页模板的实现一致，只是多页模板中包含多个`<section data-role="page">`的页面容器。

增加“第二个页面”的代码片段如下：

```
<section id="page_second" data-role="page" data-title="第二个页面">
  <header data-role="header">
    <h1>第二个页面</h1>
  </header>
  <div class="content" data-role="content">
    点击这里，返回 <a href="#page_MultiPageDemo">演示</a> 页面。
  </div>
</section>
```

(2) 在第一个页面中添加指向“第二个页面”的超级链接：

```
<a href="#page_second">第二个页面</a>
```

在图3-1左图中点击“第二个页面”链接，将得到如图3-1右图所示的页面。

jQuery Mobile的超级链接有别于其他常见的Web开发技术，详情可参阅3.3节。

经过整合之后，完整的多页模板页面的实现代码如代码清单3-1所示。

代码清单3-1 多页模板页面的实现代码

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
```

```
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <!--这里是多页显示的第一页-->
    <section id="page_MultiPageDemo" data-role="page">
      <header data-role="header">
        <h1>多页面模板演示</h1>
      </header>
      <div class="content" data-role="content">
        <p> 这是一个“多页面模板演示”。 <br/>
        请点击下面的超级链接，以跳转页面。 </p>
        点击这里，进入 <a href="#page_second">第二个页面</a>。
      </div>
    </section>

    <!--这里是多页显示的第二页-->
    <section id="page_second" data-role="page" data-title="第二个页面">
      <header data-role="header">
        <h1>第二个页面</h1>
      </header>
      <div class="content" data-role="content">
        点击这里，返回 <a href="#page_MultiPageDemo">演示</a>页面。
      </div>
    </section>
  </body>
</html>
```



图3-1 使用多页模板在一个网页中包含两个页面

3.2 页面标题

在前面的演示中，不管单页模板还是多页模板，页面代码<head>中的<title>部分都设置为

“跨平台移动应用”。按照HTML页面通常的呈现方式，<title>标签中的内容会被显示在浏览器标题栏上。但是在jQuery Mobile页面呈现的时候，浏览器标题栏上并不一定会显示这个标题，这是因为在jQuery Mobile多页模板页面中，每个页面的标题可能是不同的，需要单独设置。

实现页面标题时，可以在页面标签容器中设置属性data-title：

```
<section id="page_second" data-role="page" data-title="关于我们">
…
</section>
```


注意 如果页面容器中没有设置data-title属性，那么页面工具栏中的标题内容将会作为页面标题出现在标题栏的位置。

如果多页模板的第一个页面设置有页眉工具栏的标题，而之后的页面没有设置，而且也没有在页面容器中设置data-title属性，则跳转之后的多页模板页面会使用跳转前的页面的标题。

这样的处理显然容易引起混乱，所以建议开发者在每个页面容器中设置data-title属性。

在页眉工具栏中，不但会包含页面标题，通常还可以包含功能按钮或者导航按钮。有关页眉工具栏的进一步介绍，请参见第8章。

3.3 页面链接

超级链接是互联网应用中最核心的技术之一，如果没有超级链接，各个页面将不能连接在一起。同样，超级链接也是jQuery Mobile中最核心的技术之一。与其他Web开发技术不同，jQuery Mobile为超级链接增加了很多不同的特性并进行了优化，这将能够在有限的网络带宽和移动设备处理能力的约束下，很大程度地改善移动应用的用户体验。

对于网络带宽不稳定、移动设备运算能力有限的使用场景，页面刷新可能意味着使用者漫长的等待。在基于jQuery Mobile的应用中，如果没有特别声明，那么超级链接意味着一个Ajax地址，而不是其他网页开发技术中的超级链接。

下面我们来看看点击这个地址所发生的行为：

```
<a href="hello.html">Hello World</a>
```

这是一个通过相对路径方式指向的页面。当在jQuery Mobile应用中点击这个页面时，页面并不会马上刷新，而是先在后台进行页面的DOM加载和初始化，然后再将页面内容呈现在移动设备的浏览器上，这就是jQuery Mobile所使用的Ajax技术。

在其他的网页开发技术中，当点击超级链接之后，页面将马上跳转到新页面，并进行页面的刷新。

小技巧 基于jQuery Mobile的Ajax技术，使用异步预取和缓存来加快页面加载速度，使Web移动应用的操作更加流畅，详情可参见第4章。

如果页面不存在，那么基于Ajax的DOM加载与初始化将失败，此时就会出现错误提示。而在其他的网页开发技术中，则会出现来自服务器的404页面找不到错误。两种404错误的页面呈现效果如图3-2所示。



图3-2　jQuery Mobile（左图）和IIS服务器（右图）中的页面找不到错误

在图3-2中，错误提示Error Loading Page就是在DOM加载和初始化失败之后发出的。jQuery Mobile所提供的错误提示默认是英文的。如果所开发的应用需要适用于不同的国家，例如需要支持简体中文或者繁体中文，则可以对默认提示消息进行国际化处理，详情可参见第4章。

在jQuery Mobile应用中，超级链接大多以Ajax方式进行页面跳转，不过也有一些例外情况，详见表3-1。

表3-1　不使用Ajax超级链接的方式

不使用Ajax超级链接的方式	含　义
`<a href='http://catchexception.net/'>`	指向另一个域名下的页面
`<a href='somepage.htm' rel='external'>`	设置rel属性为external
`<a href='somepage.htm' data-ajax='false'>`	通过设置data-ajax属性为false。 默认为true，支持Ajax类型的超级链接
`<a href='somepage.htm' target='other'>`	在新窗口中打开这个页面

data-ajax是jQuery Mobile 1.1.0之后开始支持的新属性，这个属性可以用于超级链接标签之上，也可以用在超级链接之外的DOM容器上。data-ajax属性一旦被设置为false，DOM容器之内的所有超级链接都将不使用Ajax进行页面跳转，而是以通常所见的方式进行页面跳转。

如果这个禁用Ajax跳转的DOM容器中存在某个超级链接，并且这个超级链接将data-ajax属性设置为true，那么这个超级链接依然可以以Ajax方式进行页面跳转。

注意　在页面容器中使用data-ajax='false'的设置时，必须设置jQuery Mobile全局选项$.mobile.ignoreContentEnable为true，否则页面之内的超级链接将依然以默认的Ajax方式处理，此时即便将data-ajax属性设置为false也不起作用。

多页模板内部页面之间的超级链接和用作单页模板或不同的多页模板页面之间跳转的超级链接略有不同。多页模板的页面跳转是在一个网页文件内部的各个页面之间展开的。

如果在多页模板中支持超级链接和页面跳转，那么可以通过这样的代码片段实现：

```
<a href="#pageId">多页模板中的页面</a>
```

href所指向的地址不再是一个页面，而是多页模板中希望跳转到的目的页面的DOM容器id，并且需要在DOM容器id前面标记#符号。这种超级链接的形式非常类似于命名锚记。

小技巧　在多页模板中，通过#pageId的形式来跳转到某个页面，不需要重新加载和刷新页面，的确可以改善用户页面，随之而来的一个问题是，命名锚记应该如何处理？在HTML语言规范中，命名锚记也是使用#符号来标记的。

关于jQuery Mobile应用中使用命名锚记的方法，请参见第4章。

在多页模板中使用超级链接还有一个需要注意的地方，那就是通过超级链接跳转到某个多页模板文件时，默认是进入到第一个页面的。如果希望所跳转到的页面是多页模板中第二个或者以后的页面，则可以使用这样的代码片段实现：

```
<a href="MultiPage.html#secondPageDOMId">多页模板中的第二个页面</a>
```

特别需要注意的是，使用这个技巧从其他网页文件跳转到多页模板中某个指定页面时，多页模板的页面id一定要书写正确。如果写错了，将可能看不到错误提示，从而造成Web移动应用使用者的困惑，也将造成开发者难以定位和诊断问题所在。

在HTML网页开发中，往往有这样的场景：在HTTP服务器中设置若干默认的页面文件名，如果使用者浏览某个目录时没有输入确定的文件名，此时会自动打开这些默认文档。例如，在IIS中设置这些为默认文档，如图3-3所示。

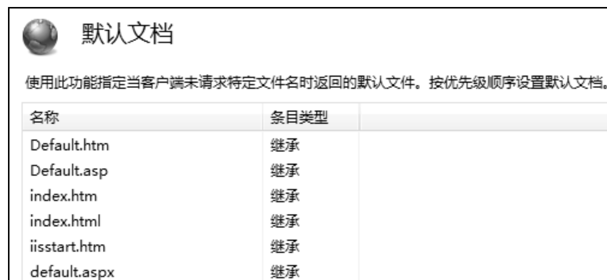


图3-3　IIS默认文档

当用户点击指向某个URL目录的超级链接时，例如

```
<a href='http://catchexception.net/about/'>…</a>
```

HTTP服务器将会检查目录中是否存在默认文档，如果存在，就将默认文档的内容发送到浏览器。对于jQuery Mobile程序，如果包含有打开目录URL地址的场景，开发者需要特别留意，一定要在目录地址的最后添加斜线/，以表示这是目录而非文件。否则，jQuery Mobile将可能通过Ajax的方式尝试打开这个URL，从而出现错误。

3.4　建立和关闭对话框

对话框可以看成一种特别的页面样式，能够显示与页面一样的内容。有别于页面，对话框不再以全屏方式呈现内容，而是在有限的范围内呈现相关的内容。图3-4分别以页面的形式和对话框的形式显示了内容和功能大致相同的两个界面。



图3-4　页面（左图）和对话框（右图）

下面继续以之前的多页模板为例进行介绍。实现对话框效果很简单，只需将多页模板中第二个页面的section容器的data-role属性值从page改为dialog即可。

为了更加接近真实的对话框样式，将之前例子中的“返回”链接调整为按钮样式。在这里，按钮样式的实现只需要在原有<a>超级链接标签中加入data-role="button"即可。关于按钮的使用，可参见第7章。

基于HTML超级链接打开对话框和打开页面的方法基本一样，不同在于页面设置上，对话框页面的data-role属性被设置为dialog。

调整后的对话框的代码如代码清单3-2所示。因为两种界面的语境不同，我们在显示文字上稍微进行了一点调整。

代码清单3-2　从对话框关闭与返回之前的页面

```
<section id="page_second" data-role="dialog">
  <header data-role="header">
    <h1>对话框</h1>
  </header>
  <div class="content" data-role="content">
    点击返回，回到上一个页面。
    <br/>
    <a href="PageTransition.html" data-role="button" data-theme="b" data-rel="back">
      返回
    </a>
    <a href="#" data-role="button" data-theme="b" onClick="jquery('page_second').dialog('close')">
      关闭</a>
   </div>
</section>
```

如果要关闭对话框，既可以使用jQuery选择器实现，也可以直接点击对话框左上方的图标来实现。

通过程序关闭对话框的代码如下：

```
$('.ui-dialog').dialog('close');
```

在对话框的方法中，只有关闭方法而没有打开方法。如果要通过JavaScript程序打开一个对话框，通常需要创建一个超级链接，并通过设置CSS样式为display:none使得这个超级链接不可见，然后通过程序模拟点击这个超级链接实现。

下面的代码将一个不可见的超级链接指向对话框#page_second上：

```
<a id='lnkDialog' href="#page_second" data-rel="dialog" data-transition="pop"
style='display:none;'></a>
```

当通过下面的JavaScript程序触发这个超级链接时，则会打开对话框：

```
$("#page_second").click();
```

注意　jQuery Mobile提供页面跳转的方法$.mobile.changePage()通常只用于页面操作而不可以直接用于打开对话框。如果想使用这个方法打开对话框，则需要添加针对对话框的参数role: 'dialog'，示例代码如下：

```
$.mobile.changePage('#page_second, {transition: 'pop', role: 'dialog'});
```

在页面和对话框中，可以通过在超级链接按钮上设置data-rel属性为back而实现页面的“后退”功能：

```
<a href="PageTransition.html" data-role="button" data-rel="back">
```

注意　在实际开发过程中，建议开发者最好能够在设置data-rel="back"之外，依然设置href属性值为上一个页面的地址。这是由于如果移动设备浏览器不能很好地支持data-rel属性，至少可以跳转到超级链接的href所指向的页面。

3.5　切换方式

不管是页面还是对话框，在呈现的时候都可以设定其切换方式，以改善用户体验，这可以通过在链接中声明data-transition属性为期望的切换方式来实现。

实现页面切换的代码如下：

```
<a href="#page_second" data-transition="slideup">自下向上切换页面</a>
```

当前，jQuery Mobile支持的页面切换方式包括10种，如表3-2所示。

表3-2　页面切换方式

切换方式	data-transition属性值
横向幻灯方式	slide
自上向下幻灯方式	slideup
自下向上幻灯方式	slidedown
中央弹出	pop
淡入淡出	fade
旋转弹出	flip
横向翻转	turn
缩小并以幻灯方式切换	flow
淡出方式显示，横向幻灯方式退出	slidefade
无动画效果	none

注意　旋转弹出等一些效果在Android早期版本中支持得不是很好。旋转弹出特效需要移动设备浏览器能够支持3D CSS，但是早期Android操作系统并不支持这些。

第4章

页面高级话题

4

使用jQuery Mobile开发的Web移动应用与传统网页的使用场景不尽相同。移动网络可能不够稳定，移动设备浏览器处理网页的速度比PC更慢，所以在进行Web移动应用开发时，JavaScript的实现方式也不同于传统的网页程序。这些区别很多是源于性能的考虑，在开发时需要格外留心。

在这一章中，我们将会了解到：

- 初始化事件响应；
- 通过预取和缓存改善用户体验；
- 命名锚记的实现；
- 通过JavaScript或HTML5 Web Storage实现页面间的参数传递；
- 页面加载消息；
- 离线应用。

4.1 初始化

在基于jQuery所开发的应用中，页面初始化是指页面下载完成、DOM对象被加载到浏览器之后触发的初始化事件，这个初始化操作是通过`$(document).ready()`事件实现的。

初始化HTML文档与初始化某个特定的jQuery Mobile页面是有区别的。因为在多页模板的页面中，一个网页文件可能包含了多个不同的页面，所以基于jQuery Mobile的Web应用的初始化事件比单纯的基于jQuery的Web应用更加丰富。

以一个包含多个页面的多页模板为例，`$(document).ready()`事件会在所有DOM对象加载完成后触发，整个HTML网页文件只触发一次，而不管它是否为多页模板。在实际的应用中，往往需要针对多页模板中的不同页面执行不同页面级别的初始化。当第一次呈现每个页面时，都将执行一次`pageinit`初始化事件。

此外，在启动jQuery Mobile的时候，会触发`mobileinit`事件。在jQuery Mobile中，这3种初始化事件是有所区别的，具体如表4-1所示。

表4-1 jQuery Mobile初始化事件

事　件	含　义
`mobileinit`	启动jQuery Mobile时触发此事件

（续）

事　　件	含　　义
$(document).ready()	HTML页面DOM对象加载完成时触发此事件
pageinit	初始化完成某个页面时，触发此事件

这些初始化事件的触发顺序如下所示。

(1) 首先触发mobileinit。

(2) 然后触发$(document).ready()。

(3) 每当第一次打开某个页面时，均触发pageinit事件。打开第一个页面时，会触发其pageinit事件。当跳转到第二个页面时，会触发第二个页面的pageinit事件。

需要注意的是，mobileinit、$(document).ready()和pageinit只能触发一次。如果从当前HTML文件的另一个页面模板跳转回之前已经访问过的页面，则不会重复触发初始化事件。

如果希望每次呈现页面时都能够触发初始化事件，则可以将这样的函数绑定到pageshow事件。或者，可以使用trigger()函数触发特定的事件。例如，

```
<a href="#" onClick="$(document).trigger('mobileinit')">手动触发mobileinit事件</a>
```

就能够实现多次触发初始化事件的目的。

除了上面介绍的3个事件外，还有onload事件。当所有相关内容（包括图片）加载完成时，会触发onload事件。因为受到图片等内容的影响，onload事件的触发时间比较晚。虽然在页面开发中也会用到onload事件，但在jQuery Mobile开发中，主要使用的是mobileinit、$(document).ready()和pageinit这3种初始化事件。

代码清单4-1演示了mobileinit、$(document).ready()和pageinit这3种事件的绑定方式。在执行过程中，将分别触发不同的事件函数。

代码清单4-1　事件触发示例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script>
      $(document).ready(function(e) {
        alert("document.ready被触发。");
      });
      $(document).live("mobileinit", function () {
        alert('mobileinit事件被执行');
      });
      $(document).delegate("#page_MobileInit0", "pageinit", function(){
        alert('page_MobileInit0页面的pageinit初始化被执行');
      });
       $(document).delegate("#page_MobileInit0", "pageshow", function(){
        alert('page_MobileInit0页面的pageshow初始化被执行');
```

```
      });
      $(document).delegate("#page_MobileInit1", "pageinit", function(){
        alert('page_MobileInit1页面的pageinit初始化被执行');
      });
    </script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="page_MobileInit0" data-role="page">
      <header data-role="header">
        <h1>页面事件</h1>
      </header>
      <div class="content" data-role="content">
        <p> jQuery Mobile页面初始化是通过mobileinit、$(document).ready()以及pageinit实现的<br />
        <a href="#" onClick="$(document).trigger('mobileinit')">手动触发mobileinit事件</a><br />
        进入<a href="#page_MobileInit1">下一页</a><br /></p>
      </div>
    </section>

    <section id="page_MobileInit1" data-role="page">
      <header data-role="header">
        <h1>页面事件</h1>
      </header>
      <div class="content" data-role="content">
        <p> jQuery Mobile页面初始化是通过pageinit实现的<br />
        <a href="#page_MobileInit0">返回</a></p>
      </div>
    </section>
  </body>
</html>
```

按照顺序，分别呈现如下消息。

(1) 启动jQuery Mobile，触发mobileinit事件，如图4-1所示。



图4-1 触发mobileinit事件

(2) DOM对象加载完成，`$(document).ready()`被触发，如图4-2所示。



图4-2　`$(document).ready()`被触发

(3) 第一个页面的`pageinit`事件被触发，如图4-3所示。



图4-3　`pageinit`事件被触发

(4) 页面中的`pageshow`事件被触发，如图4-4所示。



图4-4　pageshow事件被触发

(5) 进入第二个页面后，执行第二个页面的pageinit事件，如图4-5所示。



图4-5　pageinit事件被触发（第二个页面）

(6) 从第二个页面返回第一个页面。因为第一个页面在之前已经执行过pageinit初始化，所以这个页面的pageinit初始化不会被重复触发。当页面被呈现的时候，这个页面的pageshow事件被触发，如图4-6所示。



图4-6　pageshow事件被触发（第二个页面）

对于已经执行完mobileinit或者pageinit初始化事件的页面而言，初始化事件不可以重复执行，但是可以通过执行$(document).trigger()函数再次触发特定的初始化事件。

图4-7展现了通过$(document).trigger('mobileinit')触发mobileinit事件的场景。



图4-7　通过trigger()函数再次触发mobileinit事件

4.2 通过预取和缓存改善页面访问速度

使用单页模板时，页面在加载之后直接被呈现出来。当从一个页面跳转到下一个页面的时候，就需要重新到服务器请求新的页面内容，因此单页模板在页面跳转的时候，用户会感到页面略有停顿。使用多页模板，可以改善页面跳转之间的流畅性，但是多个页面要一次性下载，所以下载时间变长，用户体验也会受到影响。

如果能够有一种实现方式，既可以实现页面快速下载和呈现，也可以保证页面跳转和切换过程的流畅性，那将可以带来更好的用户体验，而预取（prefetch）技术就可以帮助开发者实现这样的目的。

在基于预取技术开发的应用中，当第一个页面的DOM对象加载完成之后，jQuery Mobile会对标记data-prefetch的链接地址进行预取操作。预取操作的详细过程是，在用户浏览一个页面的时候，浏览器开始下载和缓存标记为预取的页面，并将预取所指向的页面内容缓存到DOM中。这样当页面跳转到预取页面的时候，因为预取页面的内容已经加载到DOM中，就可以直接呈现出DOM缓存的页面内容。通过使用预取技术，第一个页面的呈现速度以及到后续页面的跳转速度会比单页模板更快，而下载和呈现第一个页面的速度将比打开和呈现一个多页模板网页更快。另外，经过预取技术优化的页面跳转也更少被阻塞，用户体验更加流畅。

标记预取的链接地址样式如下所示：

```
<a href="PrefetchPage.html" data-prefetch>跳转到Prefetch页面</a>
```

当页面的'pagecreate'事件触发之后，jQuery Mobile会自动下载这些预取链接所指向的页面的内容，并将其加载到DOM中。

当页面跳转到预取页面时，如果内容已经预取完成，那么在页面跳转过程中将不再显示加载消息。这与多页模板的用户体验非常类似，页面直接从前一个页面跳转到下一个页面，中间不会呈现加载消息。这是因为在预取操作执行的过程中，预取链接所指向的页面的DOM已经被加载到当前页面的DOM中。既然DOM都已经加载完成了，那么页面跳转自然顺畅地跳转到后一个页面了。

这个预取特性比较适用于页面内容连续而各个页面加载内容比较多的场景，比如具有分页设计的新闻、小说、相册或者幻灯。在阅读新闻时，当读者阅读第一页的时候，jQuery Mobile预取下一页内容。当用户点击“下一页”链接时，这个页面的内容已经在后台加载完成，几乎不需要等待就可以直接跳转到下一页。

小经验 使用预取功能时，不建议给所有链接都添加data-prefetch属性，因为过多的data-prefetch属性将会导致移动设备需要预取的页面数量过多，加载的DOM对象过大。过大的DOM对象将可能会消耗大量手机内存，从而导致一些移动设备浏览器运行缓慢，甚至崩溃。

为了有效节省移动设备浏览器的内存资源，对于没有标记为缓存的页面，在访问到下一个页面或者被预取的页面时，前一个页面的DOM对象会被清理掉。也就是说，假如用户从第一个页面跳转到下一个经过预取的页面，再重新返回第一个页面时，第一个页面的内容已经被浏览器清理掉了，需要从服务器重新下载。

如果依然希望之前的页面能够缓存在浏览器中，而不被清理掉，可以在相应的DOM对象上标记data-dom-cache="true"，例如<div data-role="page" id="CachePage" data-dom-cache="true">…</div>。

这里的DOM对象清理对于多页模板是无效的，只是会对通过Ajax方式进行的页面预取操作产生影响。对于多页模板，通常不需要设置缓存或者预取，因为多页模板的所有内容在下载之后就全部加载到了浏览器DOM中。

小经验　除了通过标记data-dom-cache属性为true来缓存页面外，开发者还可以通过使用HTML5的离线应用功能将页面内容缓存在本地。经过离线应用处理的页面，不但加载速度更快，甚至在移动网络不稳定的环境下也可以进行页面操作，详情可参考4.6节。

代码清单4-2演示了包含预取链接的单页模板，其中声明了缓存，这样当从下一个页面返回到这个页面的时候，页面内容被从DOM对象直接加载而不需要重新从服务器下载。

代码清单4-2　包含预取链接的单页模板页面

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="GBK">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <div id="page_PageTransition" data-role="page" data-dom-cache="true">
      <header data-role="header">
        <h1>预取页面处理</h1>
      </header>
    <div class="content" data-role="content">
      <p>这段演示将呈现采用与不采用预取技术的两种页面切换方式。</p>
      <a href="PrefetchPage01.html" data-prefetch>采用预取技术的页面</a><br />
      <a href="PrefetchPage02.html" rel="external" >传统的页面跳转实现</a></div>
    </div>
  </body>
</html>
```

代码清单4-3演示了被预取链接所指向的页面代码示例。

代码清单4-3　预取链接所指向的页面代码示例

```
<section id="page_PageTransition2" data-role="page">
  <header data-role="header">
    <h1>页面跳转</h1>
  </header>
  <div class="content" data-role="content">
    <p>跳转到经过预取的页面</p>
  </div>
</section>
```

代码清单4-3并不是代码片段，而是一个完整的预取页面内容。这个预取链接所指向的页面缺少了很多传统HTML页面开发中所必需的元素，例如jQuery库、jQuery Mobile库和CSS文件、head和body标签等。正因为如此，预取链接所指向的页面内容更加精简，尺寸更小，下载与加载速度也更快。

图4-8是经过前一页面跳转进入的预取页面。



图4-8　跳转到预取页面

值得注意的是，预取页面不可以直接打开。如果直接打开预取页面，页面会由于没有经过处理而呈现异常，示例图如图4-9所示。此时既不会加载jQuery和jQuery Mobile的JavaScript库与CSS，也不会加载jQuery Mobile所定义的主题样式和色版。由于页面缺少字符集的定义，中文呈现为乱码。由于缺少对meta标签中viewport属性的定义，页面显示比例很小以至于难以阅读。这样的场景是开发者要特别注意的。



图4-9　直接打开预取页面时，内容和版式出现混乱

如果将前面的预取页面改为单页模板方式，则会是代码清单4-4所示的结构。

代码清单4-4　未采用预取设计的页面代码示例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="page_PageTransition2" data-role="page">
      <header data-role="header">
        <h1>页面跳转</h1>
      </header>
      <div class="content" data-role="content">
        <p>跳转到传统的jQuery Mobile页面</p>
      </div>
    </section>
  </body>
</html>
```

运行上述代码，得到的运行结果如图4-10所示。

这个单页模板页面中包含有完整的HTML5页面信息，包括head标签、body标签、jQuery与jQuery Mobile库所需的JavaScript和CSS等相关文件。而之前，由预取链接所指向的页面统统没有这些内容。



图4-10 跳转到传统页面

此外，由于这个页面包含完整的jQuery和jQuery Mobile库、CSS、完整的HTML代码结构，所以可以直接被浏览器打开，而且不会出现异常。

不论这个单页模板页面是由前一个页面跳转过来还是直接打开的，所呈现的效果都是一致的。这样的页面由于需要加载JavaScript库以及CSS文件，比经过预取设计的页面的打开速度慢一些。

4.3 命名锚记

在HTML语法中，命名锚记用于标记页面中的特定位置。当点击指向命名锚记的超级链接时，页面将跳转到命名锚记的位置。命名锚记使用标签`<a name="anchor">命名锚记</a>`的形式来设置，可以使用`<a href="#anchor">连接到锚</a>`的形式将链接地址指向命名锚记。

默认情况下，jQuery Mobile自动通过Ajax方式处理链接点击请求，而HTML语法定义的命名锚记在jQuery Mobile不可以直接使用，示例代码如代码清单4-5所示。如果没有对命名锚记进行特定处理，则点击包含有命名锚记的链接后，不会出现任何跳转。我们可以使用一些变通方式来实现HTML中链接地址锚的使用。

代码清单4-5 HTML语法定义的命名锚记

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="page_Anchor01" data-role="page">
      <header data-role="header">
        <h1>命名锚记</h1>
      </header>
      <div class="content" data-role="content">
        <p><a href="#anchor">点击这里</a>，移动到Hello Anchor命名锚记上。</p>
        <br/>
        <br/>
          此处省略页面中的一些文字和内容。如果填充大量文字，链接和命名锚记不在同一屏中，则可以看到
            点击效果。
        <br/>
        <div style="height:500px;"></div>
        <p><a name="anchor" id="anchor">Hello Anchor</a></p>
      </div>
    </section>
  </body>
</html>
```

运行上述代码，得到的结果如图4-11所示。



图4-11　包含命名锚记的页面，但是命名锚记已不能用

设置ajaxLinksEnabled(false)，可以禁用jQuery Mobile的Ajax特性，正常使用命名锚记。但是一般我们不建议这样做，这会由于使用Ajax特性而造成异常，并且多页模板中各个页面之间的跳转或者预取页面的页面间的跳转也会出现异常。

4.3.1 在单页模板中实现命名锚记

在单页模板中实现命名锚记的一种方式是，通过捕获click或vclick事件以JavaScript函数解析用户操作行为，模拟实现命名锚记的跳转。简单地说，这并不是真正实现命名锚记，而是模拟实现命名锚记的效果。

既然jQuery Mobile中不能直接使用命名锚记，就研究一下是否存在其他变通实现的可能。在这样的变通实现中，首先需要在指向命名锚记的超级链接中绑定click和vclick事件处理函数。指向命名锚记的链接也需要标记有特定CSS定义，例如class='scroll'。被指向的命名锚记不再通过…的形式标记，因为click和vclick的事件处理函数所定位的命名锚记地址可以指向特定的DOM id所在的位置，而不再是命名锚记的位置。当点击特定的链接时，会触发click或vclick事件处理函数，这些事件处理函数会通过silentScroll()方法滚动至命名锚记的位置。

这个实现是一个变通的方法，看起来有点抽象。在这个实现中，既实现了命名锚记的功能，又保证了兼容于jQuery Mobile的Ajax页面链接可以正常运行。代码清单4-6展示了使用silentScroll()方法模拟命名锚记功能的JavaScript脚本。

4

代码清单4-6 绑定click和vclick事件，以处理单页模板命名锚记

```
<script type="text/javascript">
  $(document).ready(function(){
    $('a.scroll').bind('click vclick', function(ev) {
      var target = $($(this).attr('href')).get(0).offsetTop;
      $.mobile.silentScroll(target);
      return false;
    });
  });
</script>
```

在上述代码中，我们在标记为class="scroll"的超级链接标签<a>上绑定click和vclick事件。点击超级链接标签…，就会触发事件绑定函数。在函数内部，解析超级链接地址，找到href属性的内容。这个href属性的内容就是将要跳转到的DOM对象id名称。然后，获得这个DOM对象距离页面顶端的偏移像素值，再调用silentScroll()方法将屏幕滚动到指定位置。

需要注意的是，虽然实现的效果和命名锚记是一样的，但是指向的位置并不一定是命名锚记。在这个实现方法中，既然命名锚记已不再可用，那么代码中target变量所指向的位置也就可以是各种HTML DOM对象。例如在之后的代码示例中，命名锚记被指向到：

```
<h3 id="anchor1">当前页面的命名锚记</h3>
```

显然，这个地址并不是通常命名锚记的标记，而是一个三级标题。

在事件响应程序的最后，需要使用return false来结束程序，这样将屏蔽掉jQuery Mobile默认的Ajax处理方式。否则，后续的事件处理程序将继续执行。在这样的事件响应程序中，以return

false结束相当于分别调用event.preventDefault()和event.stopPropagation()方法。这两个方法的功能有所不同，具体如下所示。

- event.preventDefault()：通知浏览器不再触发与事件绑定的默认动作。
- event.stopPropagation()：通知浏览器当前DOM对象的事件处理程序执行之后，事件不再进行分发和处理。

完整的页面代码如代码清单4-7所示。

代码清单4-7　单页模板命名锚记实现代码示例

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type="text/javascript">
      $(document).ready(function(){
        $('a.scroll').bind('click vclick', function(ev) {
          var target = $($(this).attr('href')).get(0).offsetTop;
          $.mobile.silentScroll(target);
          return false;
        });
      });
    </script>
  </head>
  <body>
    <div data-role="page" id="firstPage">
      <div data-role="header">
        <h1>jQuery Mobile命名锚记</h1>
      </div>
      <div data-role="content">
        <a class="scroll" href="#anchor1">当前页面命名锚记</a>
        <div style="height:500px;"></div>
          <h3 id="anchor1">当前页面的命名锚记</h3>
        <p>
          在点击当前页面指向命名锚记anchor1的超级链接时，触发绑定在a.scroll上的click以及vclick事件
            函数。此时不再处理jQuery Mobile默认的Ajax请求，而是根据事件函数的定义跳转到id="anchor1"
            的标签位置，模拟实现了命名锚记的效果。
        </p>
      </div>
    </div>
  </body>
</html>
```

4.3.2　在多页模板中实现命名锚记

多页模板下命名锚记的实现方式与单页模板基本一样，不同之处在于单页模板的命名锚记跳

转为当前页面内，而多页模板需要跳转到指定页面的命名锚记位置。这样，多页模板中超级链接指向的命名锚记地址需要增加页面id，使用时首先解析跳转的目标页面id以及跳转到的命名锚记目标对象DOM id。

在代码清单4-8中，当触发绑定在超级链接上的click或vclick事件时，JavaScript将会解析href属性值#secondPage-anchor2。该属性值由两个部分构成：前半部分secondPage为页面id，在changePage()函数中用于页面跳转；后半部分anchor2为命名锚记DOM对象id，将被传入silentScroll()函数中，模拟命名锚记实现跳转。两个部分通过分隔号“-”进行分割。当触发click或者vclick的事件响应函数时，会将href属性值所包含的两个部分解析出来，再分别使用。

代码清单4-8 绑定click和vclick事件处理，以处理多页模板命名锚记

```
<script type="text/javascript">
  $(document).ready(function(){
    $('a.other-page-link').bind('click vclick', function(ev) {
      var href = $(this).attr('href');
      var parts = href.split("-");
      var page = parts[0];
      var id = "#"+parts[1];
      $.mobile.changePage($(page));
      var target = $(id).get(0).offsetTop;
      $.mobile.silentScroll(target);
      return false;
    });
  });
</script>
```

一个值得注意的细节是，有别于单页模板，多页模板中的超级链接地址被解析后，指向命名锚记的DOM对象id没有包含“#”。在解析之后，获取了所要跳转的页面地址。命名锚记处理程序通过将“#”和页面Ajax地址拼接在一起，然后再通过调用changePage()函数实现页面跳转。

另一个需要注意的细节是，绑定和处理单模板页面命名锚记的事件函数与多页面模板命名锚记是不同的，通常它们的处理事件会绑定到不同的CSS类上。这是因为不同的命名锚记的处理程序是不同的，多页模板中的命名锚记处理程序比单页模板的处理程序多了一个步骤，增加了解析页面的Ajax地址。

例如，在单页模板中，会将命名锚记处理程序绑定在：

```
$('a.scroll').bind('click vclick', function(ev) {
  ...
});
```

而在多页模板代码示例中，会将命名锚记处理程序绑定在：

```
$('a.other-page-link').bind('click vclick', function(ev) {
  ...
});
```

多页模板中，命名锚记的完整实现代码如代码清单4-9所示。标记Page对象和模拟命名锚记

对象的分隔符为-，在事件绑定程序中会按照这个分隔符进行解析。同时在命名锚记超级链接中，也会使用同样的分隔符来连接Page对象id和命名锚记标记id。JavaScript的链接地址解析规则和HTML中超级链接href属性的书写规则应该是一致的，否则将无法解析。

代码清单4-9　多页模板命名锚记实现代码示例

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type="text/javascript">
      $(document).ready(function(){
        $('a.other-page-link').bind('click vclick', function(ev) {
          var href = $(this).attr('href');
          var parts = href.split("-");
          var page = parts[0];
          var id = "#"+parts[1];
          $.mobile.changePage($(page));
          var target = $(id).get(0).offsetTop;
          $.mobile.silentScroll(target);
          return false;
        });
      });
    </script>
  </head>
  <body>
    <div data-role="page" id="firstPage">
      <div data-role="header">
        <h1>jQuery Mobile命名锚记</h1>
      </div>
      <div data-role="content">
        <a class="other-page-link" href="#secondPage-anchor2">多页模板命名锚记</a>
      </div>
    </div>
    <div data-role="page" id="secondPage">
      <div data-role="header">
        <h1>第二个页面</h1>
      </div>
      <div data-role="content">
        <h3 id="anchor2">多页模板命名锚记</h3>
        <p>
          有别于单模板页面中的命名锚记处理。在处理跳转到另一页面中的命名锚记时，click与vclick事件函数
            对链接地址进行解析，按照约定方式将地址解析为"页面id-标签id"的形式。例如，#secondPage-anchor2
            表示secondPage页面中标签id为anchor2的位置为命名锚记的目标地址。
        </p>
      </div>
    </div>
  </body>
</html>
```

图4-12和图4-13是在多页模板环境下命名锚记代码的执行效果。首先，点击多页模板命名锚记链接之前，呈现的界面如图4-12所示。



图4-12 多页模板中包含有命名锚记的起始页面

点击命名锚记后，将跳转到第二个页面，并定位到命名锚记所在的页面位置，如图4-13所示。

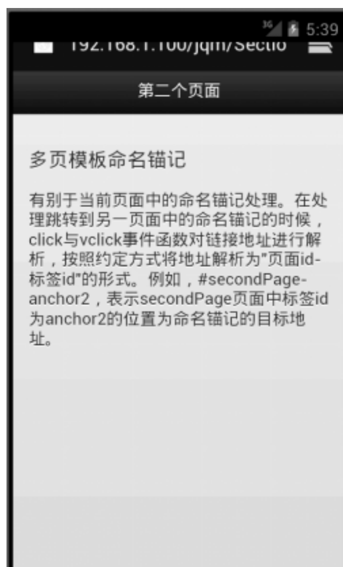


图4-13 定位到多页模板页面中的命名锚记

4.4　页面间参数传递

在一些环境下，我们需要在不同页面间实现数据传递。在基于HTTP服务器的开发中，我们最常用的一种实现方式是，前一个HTML页面通过POST和GET方式将需要传递的内容提交到HTTP服务器的处理程序，服务器再根据传入的参数将下一个页面准备好，将页面发送回浏览器，然后再在浏览器中呈现出来。在Web移动应用的开发场景中，这样的传递方式依然是最普遍的页面间参数传递方式之一。如果页面传递经由服务器进行处理，则页面间参数的传递方法和传统的HTML页面参数的处理方法是一致的。

在多页模板的各个页面之间进行参数传递时，这个方法显然就不够用了。因为多页模板的页面间切换可以不需要与服务器进行通信而直接实现。对于应用开发而言，页面间参数传递是一种常见的开发场景。

在多页模板中，通常有以下3种方法来实现页面间的参数传递。

- GET方式：在前一个页面生成参数并传入下一个页面，然后在下一个页面中进行GET内容解析。
- 通过HTML5的Web Storage进行参数传递。
- 建立当前页面的变量，在前一个页面将所需传递的参数内容赋值到变量中，在后一个页面从变量中将参数取出使用。这种方式作为多页模板的参数传递，各个页面的传入参数不同，则会表现出程序灵活性较弱的局限，所以我们将不会重点介绍这个方法。

下面我们将重点介绍前两种方法。

4.4.1　通过JavaScript实现参数传递

基于HTTP GET方式的参数传递，是通过HTTP GET的方式，将需要传递的参数附加在页面跳转的URL上。

在这样的方式下，前一个页面将参数值添加在URL地址内，后一个页面则从这个URL地址中将相应参数值解析出来，并将相应参数赋值到相应JavaScript变量上，以实现参数传递。

生成HTTP GET方式的URL地址和进行参数赋值的代码结构通常会是这样的形式：

```
<a href="?parameter=4321#PageID"rel="external">页面链接</a>
```

在基于GET方式进行参数传递的时候，参数定义在前，而Ajax指向的页面DOM对象id在后。在前面这段示例代码中，参数传递数值4321在前，而跳转到的#PageID这个页面的Ajax页面信息放在参数值之后。

另一个需要标记的内容是，注明访问的页面形式为外部链接形式rel="external"，否则页面间的参数传递将无法正常执行。

此外，后一个页面在接受来自前一个页面的参数传递时，可以通过正则表达式解析以问号?开始的部分。问号通常是URL页面和参数之间的分割符号，问号之前为页面地址，问号之后为参

数部分。将参数部分的内容解析出来，就可以获得相应的参数名称和内容。解析来自前一个页面传递参数内容的代码片段如下所示：

```
function getParameterByName(name) {
  var match = RegExp('[?&]' + name + '=([^&]*)').exec(window.location.search);
  return match && decodeURIComponent(match[1].replace(/\+/g, ' '));
}
```

小经验 大部分HTTP URL地址由7部分组成：协议名、服务器名称、域名、端口号、路径名、文件名以及参数，例如这样的地址：http://servername.domainname.net:8080/v3/index.html?parameter=HelloWorld。

由于HTTP协议的默认端口号为80，HTTPS的默认端口号为443，所以如果使用默认端口号，那么端口号就不会被单独写出来。

这种参数处理方式在大多数支持HTML 4和HTML5的浏览器环境下可以正常运行，移动设备的兼容性会比较好。

整体的代码示例如代码清单4-10所示。

代码清单4-10 以GET方式实现页面间参数传递

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type="text/javascript">
      function getParameterByName(name) {
        var match = RegExp('[?&]' + name + '=([^&]*)').exec(window.location.search);
        return match && decodeURIComponent(match[1].replace(/\+/g, ' '));
      }

      $('#page_Parameter1').live('pageshow', function(event, ui) {
        alert('第二个页面的参数：' + getParameterByName('parameter'));
      });
    </script>
  </head>
  <body>
    <section id="page_Parameter0" data-role="page">
      <header data-role="header">
        <h1>页面参数传递</h1>
      </header>
      <div class="content" data-role="content">
```

```
        <p>传递参数进入下一页，以Alert方式显示参数内容。<br />
        传递参数进入<a href="?parameter=4321#page_Parameter1" rel="external">下一页</a><br />
        </p>
      </div>
    </section>

    <section id="page_Parameter1" data-role="page">
      <header data-role="header">
        <h1>页面参数传递</h1>
      </header>
      <div class="content" data-role="content">
        <p>通过Alert显示来自前一个界面的参数。<br />
        <a href="#page_Parameter0">返回</a></p>
      </div>
    </section>
  </body>
</html>
```

在示例代码中，读取参数和通过`alert`消息框显示出来的JavaScript被绑定在第二个页面的`pageshow`事件上。当呈现页面时，会自动将参数信息呈现出来。

当第二个页面接收到来自上一个页面的参数信息时，则会呈现图4-14所示的效果。



图4-14　以HTTP GET方式通过JavaScript实现参数传递

4.4.2　通过HTML5 Web Storage特性实现参数传递

Web Storage特性是HTML5中定义的基于浏览器的存储方法，通常包含两部分——`sessionStorage`和`localStorage`。`sessionStorage`是将存储内容以会话（session）的形式存储在浏览器中，以实

现页面间通信。由于是会话级别的存储，当浏览器关闭之后，sessionStorage中的内容将会消失。localStorage是基于持久化的存储，类似于传统HTML开发中cookie的使用。除非主动声明删除localStorage中的内容，否则将不会删除。

注意　一些移动设备浏览器可能会设置localStorage的过期时间，如果超过这个时间，localStorage所存储的内容也会被移动设备浏览器清除掉。

使用HTML5的Web Storage特性实现页面间的参数传递，就是将所需要存储和传递的参数通过Web Storage在前一页面进行存储，然后在后一个页面中读取出来而实现的。由于主流的移动设备可以支持HTML5的Web Storage特性，所以在移动设备中使用Web Storage进行参数传递将会非常方便。

参数传递的代码片段如下：

```
<a href="#page_Parameter1" onClick="sessionStorage.id=hello">下一页</a>
```

和之前的通过JavaScript实现参数传递的方法不同，使用Web Storage进行参数传递不需要再次连接服务器进行页面请求，而通过当前页面跳转就可以实现。没有了JavaScript对于HTTP GET字符串的解析，代码也更加简洁。

作为HTML5的新特性，Web Storage在不同移动设备浏览器中可能存在兼容性问题，可以使用代码清单4-11检查浏览器对Web Storage的支持状况。

代码清单4-11　检查浏览器支持Web Storage的状况

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
  <html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>浏览器兼容Web Storage检查</title>
  </head>
  <body>
    <script type="text/javascript">
      if ( window.localStorage ){
        alert("浏览器支持localStorage")
      } else {
        alert("浏览器暂不支持localStorage")
      }

      if ( window.sessionStorage ){
        alert("浏览器支持sessionStorage")
      } else {
        alert("浏览器暂不支持sessionStorage")
      }
    </script>
  </body>
</html>
```

4

注意　如果直接打开存放在本地文件系统中的页面而不是通过Web服务器来打开这个页面，将无法获得Web Storage兼容性的正确返回结果。只有访问发布在Web服务器的页面，Web Storage兼容性测试程序才能正常执行。

同样，关于Web Storage的Web移动应用必须发布在Web服务器之后，再通过浏览器访问Web服务器中的页面才能正常运行。

在Android虚拟设备中，获得`localStorage`支持情况的检测结果如图4-15所示。



图4-15　浏览器支持`localStorage`

在Android虚拟设备中，获得`sessionStorage`支持情况的检测结果如图4-16所示。



图4-16　浏览器支持`sessionStorage`

通常，在jQuery Mobile中实现页面间参数传递时，我们不使用localStorage而是使用sessionStorage。因为localStorage将内容持久化在本地，这在页面间传递参数通常是不必要的。

使用sessionStorage进行多页模板各个页面间参数传递的示例代码如代码清单4-12所示。

代码清单4-12　使用sessionStorage进行页面间参数传递

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type="text/javascript">
      $('#page_Parameter1').live('pageshow', function(event, ui) {
        alert('第二个页面的参数: ' + sessionStorage.id);
      });
    </script>
  </head>
  <body>
    <section id="page_Parameter0" data-role="page">
      <header data-role="header">
        <h1>页面参数传递</h1>
      </header>
      <div class="content" data-role="content">
        <p>传递参数进入下一页，以Alert方式显示参数内容。<br />
        传递参数进入<a href="#page_Parameter1" onclick="sessionStorage.id=4321">下一页  </a><br />
        </p>
      </div>
    </section>

    <section id="page_Parameter1" data-role="page">
      <header data-role="header">
        <h1>页面参数传递</h1>
      </header>
      <div class="content" data-role="content">
        <p>通过Alert显示来自前一个界面的参数。<br />
        <a href="#page_Parameter0">返回</a></p>
      </div>
    </section>
  </body>
</html>
```

获得参数并解析的代码包含在第二个页面的pageshow事件响应函数中。当第二个页面被显示的时候，从sessionStorage中获得id数值，并通过Alert呈现出来。相比较之前的JavaScript方式，这个代码更加简洁。

由于显示参数内容的方式与前一个范例中JavaScript实现参数传递的方式一样,所以界面也是一样的，如图4-17所示。



图4-17　基于HTML5 Web Storage实现参数传递

4.5　加载消息

加载消息通常是改善用户体验设计的一种措施。当从一个页面跳转到另一个页面的时候，如果移动网络的速度慢或者需要加载的页面尺寸过大，将会造成加载时间过长，这时通过呈现加载消息将有助于改善人机交互界面。jQuery Mobile提供了默认的加载动画提示，而页面加载失败时则会呈现Error Loading Page字样。

假如被跳转到的页面并不是预取页面或者存在于多页模板中，而是某个独立的外部链接页面，这时浏览器中首先展现的是加载消息对话框，而不是直接跳转到目标页面。

如果加载错误，则会出现加载错误消息。当跳转到某个不存在的页面时，就会触发加载错误消息，如图4-18所示。

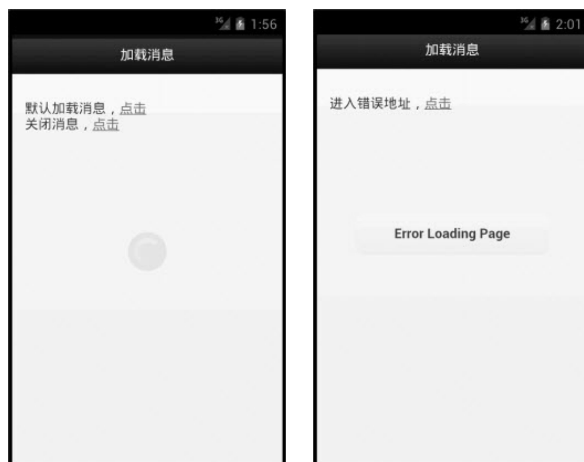


图4-18　加载消息与加载异常消息

4.5.1 自定义加载消息

默认情况下，加载的消息内容为loading，加载错误消息的内容为Error Loading Page。在进行国际化的过程中，可以定制加载消息的内容，例如使用中文呈现页面加载错误消息，这可以通过绑定mobileinit事件对loadingMessage和pageLoadErrorMessage重新赋值来实现：

```
$(document).bind("mobileinit", function(){
  $.mobile.pageLoadErrorMessage="页面加载错误";
  $.mobile.loadingMessage="页面正在加载中";
  $.mobile.loadingMessageTextVisible = true;
  $.mobile.loadingMessageTheme = "d";
});
```

图4-19为自定义页面加载消息和自定义加载错误消息的界面截图。

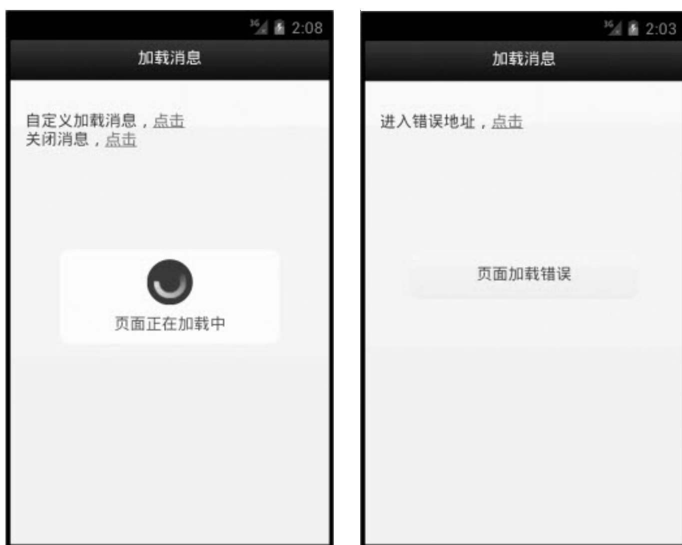


图4-19 自定义加载消息与加载错误消息

> **注意** 由于mobileinit在jQuery Mobile JavaScript库加载之后马上进行加载，所以定义pageLoadErrorMessage的JavaScript代码段需要放在引用jQuery Mobile JavaScript文件之前，否则将无法正常运行。

代码清单4-13展现了通过mobileinit初始化事件绑定实现pageLoadErrorMessage的自定义消息。需要注意的是，mobileinit的绑定必须在引用jQuery库之后、引用jQuery Mobile库之前的位置。

代码清单4-13　自定义加载消息

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script type="text/javascript">
      $(document).bind("mobileinit", function(){
        $.mobile.pageLoadErrorMessage="页面加载错误";
      });
    </script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="page_LoadMessage" data-role="page">
      <header data-role="header">
        <h1>加载消息</h1>
      </header>
      <div class="content" data-role="content">
        <p>进入错误地址，<a href="notexisting.html">点击</a><br/ >
        </p>
      </div>
    </section>
  </body>
</html>
```

同样，加载错误的消息也可以绑定在pageinit、pageshow或者onclick事件以实现定制化错误消息。下面的代码基于pageinit事件自定义加载消息：

```
$("#page_LoadMessage").live("pageinit", function(){
  $.mobile.pageLoadErrorMessage="页面加载错误";
  $.mobile. loadingMessage ="页面加载过程中";
});
```

除了能够自定义文字内容外，还可以设定加载消息是否支持动画，以及加载消息和加载错误消息的风格样式。加载消息的风格样式可以分别基于加载中的消息和加载错误的消息两类属性进行定制。常用的加载消息属性如表4-2所示，常用的加载错误消息属性如表4-3所示。

表4-2　常用加载消息属性

属　　性	含　　义
loadingMessage	设置自定义加载消息，默认为loading
loadingMessageTextVisible	如果将该属性设置为true，则表示任何情况下加载消息均会被显示
loadingMessageTheme	设置加载消息呈现风格，默认风格为a

表4-3 加载错误消息属性

属 性	含 义
pageLoadErrorMessage	当通过Ajax加载页面发生错误时呈现的页面加载错误消息，默认值为Error Loading Page
pageLoadErrorMessageTheme	设置加载错误消息的呈现风格，默认风格为e，表示淡黄色背景的消息框

4.5.2 通过JavaScript管理加载消息

在某些情况下，需要根据应用场景而触发不同的消息。例如，当在jQuery Mobile中通过延迟加载从服务器获取某个列表信息时，则可能会显示个性化信息“列表加载中…”而不是统一的“页面加载中…”消息。若想通过程序触发加载消息，可以使用$.mobile.showPageLoadingMsg()方法，示例代码如下：

```
<a href="#" onClick="$.mobile.showPageLoadingMsg('b', '显示自定义消息框', true); ">
  启动自定义消息框
</a>
```

该方法包含3个参数，具体如表4-4所示。

表4-4 showPageLoadingMsg()参数的定义

参 数	含 义
Theme	加载消息界面呈现风格，默认为a
msgText	加载消息显示文字
textonly	如果为true，则只显示文字，否则只显示图标

如果将textonly参数设置为true，将显示文字消息框，如图4-20左图所示；如果将textonly参数设置为false，将不显示文字消息框，如图4-20右图所示。



图4-20 使用showPageLoadingMsg()实现自定义消息框

如果不需要实现自定义消息，而仅仅在程序需要的时候触发加载消息框，此时使用不带有任何参数的$.mobile.showPageLoadingMsg()方法就可以了。

与页面跳转过程中的加载消息不同，通过程序触发的加载消息框是不会自行关闭的，如果没有通过程序加以控制，消息框将始终在页面上。要想关闭消息框，可以使用$.mobile.hidePageLoadingMsg()，比如：

```
关闭消息，<a href="#" onClick="$.mobile.hidePageLoadingMsg();">点击</a>
```

注意　使用JavaScript调用$.mobile.showPageLoadingMsg()并通过参数自定义消息框在jQuery Mobile 1.0和1.1.0这两个版本下的呈现是不同的。在jQuery Mobile 1.0库中，通常只能呈现默认消息框，而1.1.0则可以实现自定义消息。如果在jQuery Mobile 1.0下进行自定义消息实现，则需要封装到相对复杂的JavaScript函数中来实现。

4.6　离线应用

很多场景下（例如穿越地下通道或者漫步在群山之间），移动设备的网络并不是那么稳定或者干脆没有网络信号，这将会影响Web移动应用的用户体验。借助于HTML5的离线应用功能，开发者能够开发出即便网络不可用，页面内容只要被缓存就可以正常打开和操作的Web移动应用。图4-21演示了网络连接失败时Android模拟器中所呈现的页面，左图为网络连接中的离线应用，右图为网络中断时打不开页面的情况。

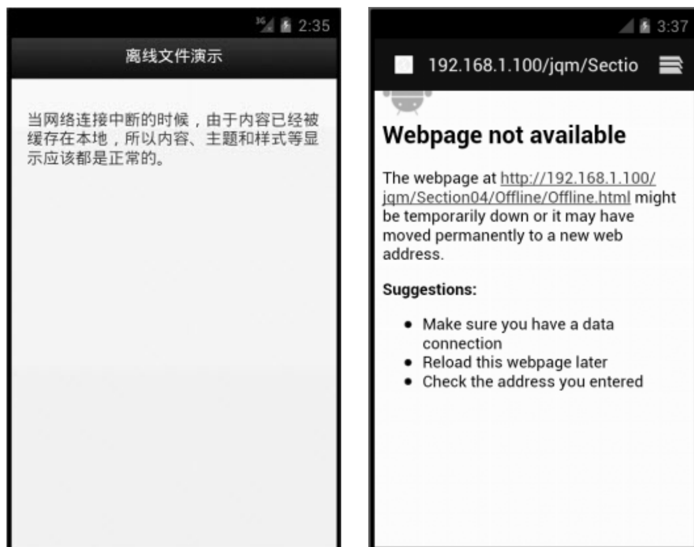


图4-21　网络连接失败时Android模拟器中所呈现的页面

小技巧 使用Android模拟器时，使用F8键可以启动与关闭网络连接。

在图4-21所示的示例中，就是使用这样的方式停止模拟器的网络连接的。仔细看，左图的右上角有3G标记，表示模拟器正在连接网络中，而右图没有3G标记，表示模拟器的网络连接被断开。

借助HTML5的离线Web移动应用功能，可以将所有需要缓存的内容缓存在浏览器中。在网络连接失败的时候，通过加载缓存中的内容，保证用户依然可以访问相应的应用。

将这个示例Web应用程序改造为离线Web应用后，即便移动设备的网络连接中断，再次刷新界面，之前缓存的内容依然可以正常显示，显示效果如图4-22所示。



图4-22 模拟器网络中断，刷新页面所呈现的离线Web应用界面

比较图4-22和图4-21，发现只有右上角的网络连接符号不同，前一个页面拥有网络连接，而后一个则没有。

离线Web应用的功能只能在HTML5中使用。开发离线Web应用与其他HTML5应用大致相同，需要如下两个步骤。

(1) 调整Web服务器以支持HTML5的离线Web应用功能。

(2) 编辑离线应用中用于缓存规则设置的.manifest文件，然后将.manifest文件集成到HTML5页面中。

4.6.1 配置Web服务器以支持离线应用

不论将Web移动应用发布在IIS服务器还是Apache服务器上，默认配置下均不支持离线应用。所以，首先需要配置Web服务器中.manifest扩展名的文件使用正确的MIME类型。

默认情况下，IIS中已经包含有.manifest扩展名所对应的MIME类型。双击扩展名，打开“编辑MIME类型”对话框，将.manifest扩展名所对应的MIME类型设置为text/cache-manifest，如图4-23所示，此时IIS就可以支持离线应用了。

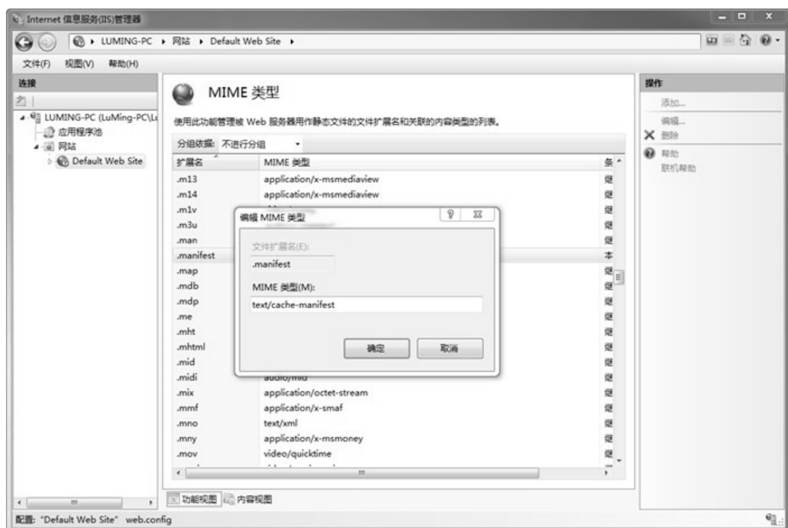


图4-23　配置.manifest文件的MIME类型

在Apache服务器中，也需要设置.manifest的MIME类型来支持离线应用。在Apache服务器中，需要在Apache配置文件httpd.conf中添加这样的配置项：

```
AddType text/cache-manifest .manifest
AddType text/cache-menifest .apache
```

Apache配置文件httpd.conf默认位于/etc/httpd/conf/httpd.conf。当httpd.conf配置文件被修改之后，重新启动Apache服务器，则这个配置就生效了。

在Red Hat Linux下，重新启动Apache HTTP服务的命令为：`service httpd restart`。

4.6.2　开发与集成离线应用

在Web移动应用中要实现HTML5的离线功能，需要完成如下3个步骤。

(1) Web服务器支持.manifest文件的MIME格式设定，这部分我们在4.6.1节中介绍过。

(2) 在网页的`<html>`标签中加入`manifest`属性，并指到.manifest文件。

(3) 定义.manifest、缓存文件清单和缓存方式。

首先，正确设置.manifest文件。在前一节中，我们介绍了在HTTP服务器中添加.manifest文件的实现方法。在此基础上，我们在HTML5网页文件中添加.manifest的定义，则可以实现离线应用。

我们可以自己定义.manifest文件的内容，但是文件格式需要为UTF-8。将.manifest文件定义添加到HTML5页面的`html`标签中，浏览器会根据.manifest文件的定义缓存离线环境下所需要使用的文件：

```
<html manifest="offline.manifest">
```

代码清单4-14演示了完成的离线应用HTML网页代码。

代码清单4-14 HTML5离线应用页面代码示例

```
<!DOCTYPE html>
    <html manifest="offline.manifest">
      <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>跨平台移动应用</title>
        <link rel="stylesheet" href="js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
        <script src="js/jquery-1.7.1.min.js"></script>
        <script src="js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
      </head>
      <body>
        <section id="page_Offline" data-role="page">
          <header data-role="header">
            <h1>离线文件演示</h1>
          </header>
          <div class="content" data-role="content">
            <p>
             当网络连接中断的时候，由于内容已经被缓存在本地，所以内容、主题和样式等显示应该都是正常的。
            </p>
          </div>
        </section>
      </body>
    </html>
```

这里.manifest文件被命名为offline.manifest，其中包含离线环境下需要用到的文件清单。基于这个.manifest文件所定义的离线文件清单，Offline.html页面、jQuery和jQuery Mobile库已经在本地缓存下来。当网络不能连接的时候，浏览器会通过HTML5的离线应用技术直接从本地调取这些文件。代码清单4-15是离线应用manifest的代码示例。

代码清单4-15 HTML5离线应用manifest的代码示例

```
CACHE MANIFEST
Offline.html
js/jquery-1.7.1.min.js
js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css
js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js
```

在离线应用的.manifest定义中，除了定义离线缓存的文件外，还可以定义不进行离线缓存的文件以及网络不通时重定向的文件名。

在.manifest文件中增加`NETWORK`部分，将可以定义必须进行网络访问的文件。例如，在offline.manifest文件下方添加如下的代码，则不缓存这几个页面：

```
NETWORK:
login.php
logout.php
```

此外，在.manifest文件中定义`FALLBACK`，可以定义网络连接不通的情况下可替代的访问资源。

第5章 弹出页面

弹出页面是jQuery Mobile 1.2.0开始支持的新特性。使用弹出页面，开发者能够快速开发出用户体验更好的移动应用。基于弹出页面，开发者可以定制浮在移动设备浏览器之上的对话框、菜单、提示框、表单、相册和视频，甚至可以集成第三方的地图组件。

在这一章中，我们将会了解到：

- 弹出页面的基本概念；
- 不同形式的弹出效果；
- 集成弹出图片、视频、地图和覆盖面板等高级特效；
- 定制弹出页面样式；
- 开发弹出页面的相关技术。

5.1 基本的弹出页面

在基于jQuery Mobile 1.1.1版本或者更早版本开发的Web移动应用中，虽然支持丰富的页面切换，但是没有提供在一个页面中弹出一个浮动页面或者对话框的功能，这在jQuery Mobile 1.2.0及其之后的版本中得以实现。

弹出页面是一个相对宽泛的概念，包括弹出对话框、菜单和嵌套菜单、表单、图片、视频、覆盖面板、地图等不同的形式。几乎所有能够用来“弹出”的页面元素，都可以通过一定方式嫁接在弹出页面这个特性上。

图5-1呈现了以弹出页面技术实现弹出对话框的效果，点击左图的“提示框”按钮，将呈现右图的弹出提示框。当使用者打开一个弹出提示框时，一个提示框将在当前页面呈现出来，而不需要跳转到其他页面。

实现弹出页面时，需要实现两个部分——弹出按钮和弹出页面。弹出按钮通常基于一个超级链接实现。在超级链接中，设置属性data-rel为popup，表示以弹出页面方式打开所指向的内容：

```
<a href="#popupTooltip" data-rel="popup" data-role="button" data-inline="true">提示框</a>
```

弹出页面部分通常是一个div的DOM容器，其中将data-role属性声明为popup，表示以弹出方式呈现其中的内容：

```
<div data-role="popup" id="popupTooltip">
  …
</div>
```



图5-1　弹出对话框的效果

和其他多页模板中打开对话框或者页面的方式一样，超级链接中href属性所指向的地址是页面DOM容器的id。当使用者点击超级链接时，则打开弹出页面。因为超级链接的data-rel设置为popup，以及页面的data-role也设置为popup，则这样的页面将以弹出页面的形式打开。

图5-1的完整代码示例如代码清单5-1所示。

代码清单5-1　基本的弹出窗口代码结构

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
        <h1>弹出页面</h1>
      </header>
      <div class="content" data-role="content"><a href="#popupTooltip"
        data-rel="popup" data-role="button" data-inline="true">提示框</a>
        <div data-role="popup" id="popupTooltip">
          <p><strong>提示：</strong></p>
```

```
            <p>提示框可以加在移动应用中，用于提醒和指示用户进行后续操作。
            </p>
          </div>
        </div>
      </section>
    </body>
  </html>
```

5.2　不同的弹出效果

很多用户界面都适合使用弹出界面来呈现，例如：

- ❑ 提示框；
- ❑ 菜单与嵌套菜单；
- ❑ 表单；
- ❑ 对话框。

5.1节已经介绍了提示框弹出页面，所以本节将不再重复介绍这部分内容，而介绍剩下的3种情况。

5.2.1　菜单与嵌套菜单

在弹出页面中，菜单有助于使用者在操作过程中选择功能或切换页面，如图5-2所示。



图5-2　弹出菜单

菜单和嵌套菜单的超级链接设计与所有其他弹出页面的超级链接按钮几乎是一样的。需要注意的是，在超级链接按钮中，增加值为popup的属性data-rel，然后将超级链接地址指向弹出菜单的DOM容器id：

```
<a href="#popupMenu" data-rel="popup" data-role="button" data-inline="true">
  菜单 - 地理分界
</a>
```

若要实现弹出菜单的功能，只要将包含有菜单的列表视图加入到弹出页面div容器中即可，具体代码如下：

```
<div data-role="popup" id="popupMenu" data-theme="a">
  <ul data-role="listview" data-inset="true" style="min-width:210px;" data-theme="b">
    <li data-role="divider" data-theme="a">地理分界</li>
    <li><a href="#">亚洲</a></li>
    <li><a href="#">非洲</a></li>
    <li><a href="#">欧洲</a></li>
    <li><a href="#">南美洲</a></li>
    <li><a href="#">北美洲</a></li>
    <li><a href="#">南极洲</a></li>
    <li><a href="#">大洋洲</a></li>
  </ul>
</div>
```

如果需要分类显示菜单，则菜单中的分类条目可以通过设置data-role属性为divider来实现。菜单分类显示的样式可以参见图5-3左图。如果菜单高度比较小，那么分类之后便于大家识别和定位。如果菜单条目很多，这个设计就不方便了。如果菜单高度超过移动设备浏览器的高度，操作菜单时还需要滚动屏幕，这样很容易误碰到菜单之外的区域，而关闭菜单。



图5-3　左图为分类菜单，右图为嵌套菜单

在菜单条目很多的场景下，使用嵌套菜单能获得更好的用户体验。在图5-3所示的示例中，两个菜单列表所呈现的菜单条目是一样的。左图将所有菜单条目自上而下列出来，右图则使用了嵌套菜单的效果。相比之下，右侧的嵌套菜单更方便进行菜单操作和定位到某个菜单条目上。

要实现嵌套菜单，可以通过在弹出页面中嵌入折叠列表。折叠列表是从jQuery Mobile 1.2.0

开始支持的，这将在第9章中详细介绍。

在将折叠列表装入弹出页面的div容器之后，通过点击弹出页面的超级链接按钮就可以打开这个嵌套菜单。

> **注意**　嵌套菜单是通过集成折叠列表实现的。与折叠列表的使用约束一样，嵌套菜单只支持一级嵌套，而不支持多级嵌套。

图5-3的完整代码示例如代码清单5-2所示。

代码清单5-2　弹出菜单和弹出嵌套菜单

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
        <h1>弹出页面</h1>
      </header>
      <div class="content" data-role="content">
        <a href="#popupMenu" data-rel="popup" data-role="button"
          data-inline="true">菜单 - 地理分界</a>
        <div data-role="popup" id="popupMenu" data-theme="a">
          <ul data-role="listview" data-inset="true" style="min-width:210px;" data-theme="b">
            <li data-role="divider" data-theme="a">地理分界</li>
            <li><a href="#">亚洲</a></li>
            <li><a href="#">非洲</a></li>
            <li><a href="#">欧洲</a></li>
            <li><a href="#">南美洲</a></li>
            <li><a href="#">北美洲</a></li>
            <li><a href="#">南极洲</a></li>
            <li><a href="#">大洋洲</a></li>
          </ul>
        </div>
        <a href="#popupNested" data-rel="popup" data-role="button"
          data-inline="true">级联菜单 - 地理分界</a>
        <div data-role="popup" id="popupNested" data-theme="none">
          <div data-role="collapsible-set" data-theme="b" data-content-theme="c"
            data-collapsed-icon="arrow-r" data-expanded-icon="arrow-d" style="margin:0;
              width:250px;">
            <div data-role="collapsible" data-inset="false">
              <h2>七大洲</h2>
                <ul data-role="listview">
```

```
                <li><a href="#">亚洲</a></li>
                <li><a href="#">非洲</a></li>
                <li><a href="#">欧洲</a></li>
                <li><a href="#">南美洲</a></li>
                <li><a href="#">北美洲</a></li>
                <li><a href="#">南极洲</a></li>
                <li><a href="#">大洋洲</a></li>
              </ul>
          </div>
          <!-- /collapsible -->
          <div data-role="collapsible" data-inset="false">
            <h2>六大洲</h2>
              <ul data-role="listview">
                  <li><a href="#">亚欧大陆</a></li>
                  <li><a href="#">非洲</a></li>
                 <li><a href="#">南美洲</a></li>
                  <li><a href="#">北美洲</a></li>
                 <li><a href="#">南极洲</a></li>
                  <li><a href="#">大洋洲</a></li>
              </ul>
            </div>
          <div data-role="collapsible" data-inset="false">
            <h2>五大洲</h2>
              <ul data-role="listview">
                  <li><a href="#">亚欧大陆</a></li>
                  <li><a href="#">非洲</a></li>
                <li><a href="#">美洲</a></li>
                <li><a href="#">南极洲</a></li>
                <li><a href="#">大洋洲</a></li>
              </ul>
           </div>
             <div data-role="collapsible" data-inset="false">
             <h2>四大洲</h2>
               <ul data-role="listview">
                  <li><a href="#">亚非欧大陆</a></li>
                  <li><a href="#">美洲</a></li>
                  <li><a href="#">南极洲</a></li>
                  <li><a href="#">大洋洲</a></li>
              </ul>
            </div>
         </div>
       </div>
     </div>
   </section>
 </body>
</html>
```

5.2.2 表单

在jQuery Mobile 1.2.0之前的表单实现中，只能在页面中嵌入表单。如果将表单嵌入在一个弹出页面中，那么表单的内容将更加突出。

和所有的HTML表单操作没有什么两样，在提交弹出表单的内容时，表单内容都可以提交到Web服务器进行进一步处理。

要实现弹出表单，只需在弹出页面的div容器中加入表单即可，示例代码如代码清单5-3所示。

代码清单5-3　弹出表单的示例代码

```
<div data-role="popup" id="popupForm" data-theme="b" class="ui-corner-all">
  <form>
    <div style="padding:10px 20px;">
      <input type="text" name="name" id="name" value="" placeholder="登录名" data-mini="true"/>
      <input type="Password" name="password" id="password" value=""
        placeholder="密码" data-mini="true"/>
      <div style="margin-top:20px;" >
        <fieldset class="ui-grid-a" data-mini="true">
          <div class="ui-block-a">
            <button type="reset" data-theme="d">取消</button>
          </div>
          <div class="ui-block-b" data-mini="true">
            <button type="submit" data-theme="a">提交</button>
          </div>
        </fieldset>
      </div>
    </div>
  </form>
</div>
```

运行代码清单5-3，获得的效果如图5-4所示。



图5-4　弹出表单

在前面这个示例中，将表单的theme色板设置为b，这是一种底色为灰色的配色。开发者可以尝试不同的主题色版，不同色版将呈现不同的配色效果。jQuery Mobile默认支持5种色板，分别

对应data-theme属性的a、b、c、d、e。开发者可以选择不同的色板以美化弹出效果，示例代码如下：

```
<div data-role="popup" id="popupForm" data-theme="b" class="ui-corner-all">
```

注意　在弹出页面表单中，需要对表单元素距离弹出页面的边界进行定义，具体代码为：<div style="padding:10px 20px;">。

在弹出页面的设计中，这个表单的边距设置是必须要注意的。否则，表单元素和弹出页面会拥挤在一起而显得局促。设置和不设置边距的呈现效果参见图5-5。如果不是弹出表单，通常不需要特别增加这样的边距设计。



图5-5　左图为设置边距的样式，右图为没有设置边距的样式

注意　未来，jQuery Mobile有可能通过增加新的CSS样式定义解决这个问题。如果开发者需要手工实现表单在弹出页面中的边距设定，最好能够根据不同屏幕分辨率使用CSS3的Media Queries技术选择不同的边距设定。

因为普通移动屏幕和高分辨率屏幕的呈现效果可能不同，通过CSS3的Media Queries技术将可以更好地应对这样的场景。

5.2.3　对话框

弹出对话框是弹出页面最常用的功能，如图5-6所示。在之前介绍的对话框页面中，往往需要从一个页面切换到对话框页面才能显示对话框内容，而基于弹出页面对话框，用户将不需要进

行页面切换就可以直接看到对话框的内容。



图5-6　弹出对话框

实现弹出对话框的方法是，声明一个div容器，并设置data-role属性为popup，然后将弹出对话框的代码装入这个弹出页面的div容器中即可。当使用者点击超级链接按钮时，打开的内容就是这个弹出对话框了。图5-6的实现代码如代码清单5-4所示。

代码清单5-4　弹出对话框代码示例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
        <h1>弹出页面</h1>
      </header>
      <div class="content" data-role="content">
        <a href="#popupDialog" data-rel="popup" data-role="button" data-inline="true">
          弹出对话框
        </a>
        <div data-role="popup" id="popupDialog" data-overlay-theme="a"
```

```
            data-theme="c" style="max-width:400px;" class="ui-corner-all">
            <div data-role="header" data-theme="a" class="ui-corner-top">
              <h1>关于弹出对话框</h1>
            </div>
            <div data-role="content" data-theme="d" class="ui-corner-bottom ui-content">
              <p>弹出页面特效是jQuery Mobile 1.2.0的新特性。</p>
              <p>弹出对话框是弹出页面特效的一种，呈现为对话框的效果。</p>
              <a href="#" data-role="button" data-rel="back" data-transition="flow"
                data-theme="b">
                确认
              </a>
            </div>
          </div>
        </div>
      </section>
    </body>
  </html>
```


通常，弹出对话框中只包含页眉标题栏和正文内容部分。在某些场景下，弹出对话框也可能包含页脚工具栏，但这并不常见。

在前面这个示例中，设置data-role属性为header的div容器所包含的内容为页眉标题栏。页眉标题栏中h1到h6标题所包含的文字将会作为标题栏的文字突出显示。

```
<div data-role="header">…</div>
```

对话框的正文被放置在data-role属性为content的div容器中：

```
<div data-role="content">…</div>
```

如果需要设置页脚工具栏，则可以将相应内容放置于data-role属性为footer的div容器中：

```
<div data-role="footer">…</div>
```

5.3　弹出页面的高级功能

图片、视频、地图和覆盖面板也是弹出页面中使用较多的技术。基于弹出页面，使用者不用切换页面就能打开不同的媒体文件，并浏览媒体内容。使用这些新特性，开发者只需编写很少的代码就能获得有趣的用户体验效果。

5.3.1　图片

在弹出图片的设计中，图片几乎占据整个弹出页面，突出而醒目地呈现在浏览器中，如图5-7所示。

简单地说，实现弹出图片的方法是将图片添加在弹出页面div容器中。

此时图片会按比例最大程度地填充整个弹出页面。如果图片的尺寸和浏览器的尺寸正好一致，那么可能因为没有可以触发关闭弹出页面的地方导致用户不方便跳转回之前的页面。因此，在弹出页面中，通常会包含一个关闭按钮，具体代码如下。

```
<div data-role="popup" id="popupPhoto" data-overlay-theme="a" data-theme="c"
  style="max-width:400px;" class="ui-corner-all">
  <a href="#" data-rel="back" data-role="button" data-theme="a" data-icon="delete"
    data-iconpos="notext" class="ui-btn-right">Close</a>
  <img class="popphoto" src="images/Penguins.jpg" alt="Colorful city"/>
</div>
```



图5-7　弹出图片

在Close超级链接按钮中，我们将属性data-iconpos设置为notext，而将data-rel属性设置为back。点击Close按钮后，页面会返回到上一个页面，也就是退出弹出页面而回到之前的页面。

在实际的使用过程中，移动设备屏幕会在水平方向和垂直方向之间切换。随着屏幕方向的变化，图片可能会超出屏幕显示范围，此时为了不遮挡图片，需要在页面加载的时候计算屏幕尺寸，并根据屏幕尺寸减去一定的边际值而重新设置弹出图片的尺寸。在代码清单5-5中，pageinit事件中设置图片的最大尺寸会比屏幕高度小50像素就是基于此。

代码清单5-5　在pageinit事件中设置图片尺寸

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type="text/javascript">
      $( document ).on( "pageinit", function() {
        $( "#popupPhoto" ).on({
          popupbeforeposition: function() {
```

```
            var maxHeight = $( window ).height() - 50 + "px";
            $( "#popupPhoto img" ).css( "max-height", maxHeight );
          }
        });
      });
    </script>
  </head>
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
        <h1>弹出页面</h1>
      </header>
      <div class="content" data-role="content">
        <a href="#popupPhoto" data-rel="popup" data-role="button" data-inline="true">
          弹出图片
        </a>
        <div data-role="popup" id="popupPhoto" data-overlay-theme="a"
          data-theme="c" style="max-width:400px;" class="ui-corner-all">
          <a href="#" data-rel="back" data-role="button" data-theme="a"
            data-icon="delete" data-iconpos="notext" class="ui-btn-right">
            Close
          </a>
          <img class="popphoto" src="images/Penguins.jpg" alt="Colorful city"/>
        </div>
      </div>
    </section>
  </body>
</html>
```

图5-8是在移动设备中水平显示弹出图片时的效果。因为屏幕比例发生变化，此时图片高度和宽度略微发生一些调整，以便于显示。

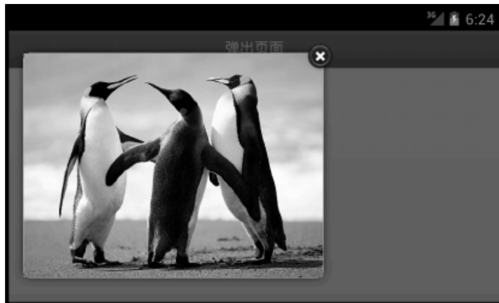


图5-8　水平方向上弹出图片的显示效果

5.3.2　视频

视频内容也可以通过弹出页面来显示，而使用者可以在包含有弹出视频的页面中浏览视频内容。

实现弹出视频页面与实现弹出图片的方式大致相同，只需要将播放视频的iframe或者embed标签的内容嵌入到弹出页面的div容器中即可，示例图如图5-9所示。

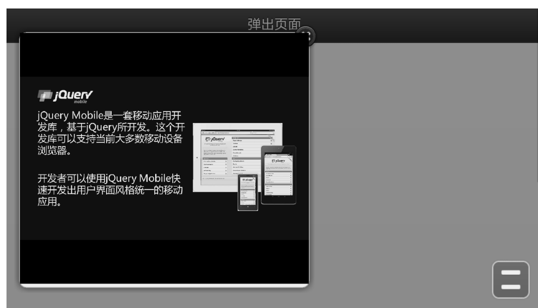


图5-9　弹出视频

通常，为了保证呈现效果足够好，开发者们建议能够设置一定的页边距，这可以通过scale()[①]函数来实现。基于用户界面设计经验，通常会保留15像素的页边距。

当移动设备发生垂直或者水平切换的时候，移动应用程序最好能够读取切换后的屏幕尺寸。如果视频播放器超出旋转之后的浏览器的边界，最好能够通过程序成比例缩放播放器。

这与之前所介绍的弹出图片中的场景类似，不同的是这需要成比例缩放，而不可以只对宽度或者高度进行缩放处理。

scale()函数的另一个功能就是根据浏览器的尺寸设置合适的弹出页面显示尺寸，其代码如代码清单5-6所示。

代码清单5-6　用于内容缩放和边距处理的scale()函数

```
function scale( width, height, padding, border ) {
  var scrWidth = $( window ).width() - 30,
    scrHeight = $( window ).height() - 30,
    ifrPadding = 2 * padding,
    ifrBorder = 2 * border,
    ifrWidth = width + ifrPadding + ifrBorder,
    ifrHeight = height + ifrPadding + ifrBorder,
    h, w;

  if ( ifrWidth < scrWidth && ifrHeight < scrHeight ) {
    w = ifrWidth;
    h = ifrHeight;
  } else if ( ( ifrWidth / scrWidth ) > ( ifrHeight / scrHeight ) ) {
    w = scrWidth;
    h = ( scrWidth / ifrWidth ) * ifrHeight;
  } else {
    h = scrHeight;
    w = ( scrHeight / ifrHeight ) * ifrWidth;
  }

  return {
    'width': w - ( ifrPadding + ifrBorder ),
```

① 有关scale()函数的相关信息，请参阅 http://jquerymobile.com/demos/1.2.0/docs/pages/popup/popup-iframes.html。

```
    'height': h - ( ifrPadding + ifrBorder )
  };
};
```

注意 scale()函数是弹出页面中经常使用的技术。尽管jQuery Mobile文档推荐使用scale()函数进行视频和地图边界设定，但是这个函数并没有包含在jQuery Mobile库或者jQuery库中，开发者可以直接将这个代码引用到所需要的页面中。

另一个需要注意的细节是，很多视频内容是通过嵌入第三方网站的iframe实现的。在进行页面初始化的时候，需要将iframe的高度和宽度设置为0。在打开视频播放器，播放器页面创建完成而未呈现在浏览器界面上的时候，重新绘制iframe的尺寸到期望的尺寸。在关闭播放器页面时，再重新设置iframe的高度和宽度为0。

代码清单5-7是对视频内容进行尺寸设定的代码示例。

代码清单5-7 通过绑定弹出页面事件设定视频播放器尺寸

```
$( document ).on( "pageinit", function() {
  $( "#popupVideo embed" )
    .attr( "width", 0 )
    .attr( "height", 0 );

  $( "#popupVideo" ).on({
    popupbeforeposition: function() {
      var size = scale( 480, 415, 15, 1 ),
      w = size.width,
      h = size.height;

      $( "#popupVideo embed" )
        .attr( "width", w )
        .attr( "height", h );
    },
    popupafterclose: function() {
      $( "#popupVideo embed" )
        .attr( "width", 0 )
        .attr( "height", 0 );
    }
  });
});
```

注意 集成视频网站内容的方式不同，通过绑定弹出页面事件进行视频播放器尺寸设定的实现方式也略有不同。如果视频播放器通过embed标签直接嵌套在弹出页面div容器中，则可以使用前面的代码。如果视频播放器通过iframe标签以内联框架的方式嵌套在弹出页面div容器中，则需要将$(> "#popupVideo embed")调整为$("#popupVideo iframe")。

弹出视频的完整页面代码如代码清单5-8所示。

代码清单5-8　弹出视频

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.1/
      jquery.mobile-1.3.1.min.css" />
    <script src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
    <script src="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type="text/javascript">
      function scale( width, height, padding, border ) {
        var scrWidth = $( window ).width() - 30,
          scrHeight = $( window ).height() - 30,
          ifrPadding = 2 * padding,
          ifrBorder = 2 * border,
          ifrWidth = width + ifrPadding + ifrBorder,
          ifrHeight = height + ifrPadding + ifrBorder,
          h, w;

        if ( ifrWidth < scrWidth && ifrHeight < scrHeight ) {
          w = ifrWidth;
          h = ifrHeight;
        } else if ( ( ifrWidth / scrWidth ) > ( ifrHeight / scrHeight ) ) {
          w = scrWidth;
          h = ( scrWidth / ifrWidth ) * ifrHeight;
        } else {
          h = scrHeight;
          w = ( scrHeight / ifrHeight ) * ifrWidth;
        }

        return {
          'width': w - ( ifrPadding + ifrBorder ),
          'height': h - ( ifrPadding + ifrBorder )
        };
      };

      $( document ).on( "pageinit", function() {
        $( "#popupVideo embed" )
          .attr( "width", 0 )
          .attr( "height", 0 );

        $( "#popupVideo" ).on({
          popupbeforeposition: function() {
            var size = scale( 480, 415, 15, 1 ),
              w = size.width,
              h = size.height;

            $( "#popupVideo embed" )
              .attr( "width", w )
              .attr( "height", h );
          },
```

```
            popupafterclose: function() {
              $( "#popupVideo embed" )
                .attr( "width", 0 )
                .attr( "height", 0 );
            }
        });
      });
  </script>
  </head>
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
           <h1>弹出页面</h1>
        </header>
      <div class="content" data-role="content"> <a href="#popupVideo"
        data-rel="popup" data-role="button" data-inline="true">弹出土豆视频</a>
        <div data-role="popup" id="popupVideo" data-overlay-theme="a"
          data-theme="c" style="max-width:480px;" class="ui-corner-all">
        <a href="#" data-rel="back" data-role="button" data-theme="a"
          data-icon="delete" data-iconpos="notext" class="ui-btn-right">Close</a>
          <embed src="http://www.tudou.com/v/NhnT3t7PSTw/&rpid=117243609&resourceId=117243609_05_
            05_99&bid=05/v.swf"
          type="application/x-shockwave-flash" allowscriptaccess="always"
          allowfullscreen="true" wmode="opaque" width="480" height="400"></embed> </div>
        <div style="height:500px"></div>
      </div>
    </section>
  </body>
</html>
```


移动设备的网络带宽可能不稳定，其分辨率也参差不齐，同时并不是每个移动设备都支持高清视频。如果能够针对移动设备分辨率的不同而加载不同尺寸的视频预览图片，将能提供更好的用户体验。很多海外视频网站（比如Facebook）能够为开发者和使用者提供不同尺寸的视频和预览图片以实现这样的优化。

以Facebook视频为例，使用Graph API可以获得同一个视频内容的不同分辨率的显示脚本。Graph API是Facebook所提供的开放API接口，下面的代码片段展示了获得id为10151026962332084的视频的基本信息：

```
https://graph.facebook.com/ 10151026962332084?fields=id,name,format&access_token={YourFacebook
Token}
```

通过Facebook Graph API调用，获得的JSON格式结果如代码清单5-9所示。

代码清单5-9　Facebook返回的视频内容

```
{
  "id": "10151026962332084",
  "from": {
    "name": "Lu Ming",
    "id": "727972083"
  },
```

```
"description": "jQuery Mobile的基本概念",
"picture": "http://vthumb.ak.fbcdn.net/hvthumb-ak-snc6/
  245287_10151026963867084_10151026962332084_24448_571_t.jpg",
"embed_html": "<object width=\"400\" height=\"224\" >
  <param name=\"allowfullscreen\" value=\"true\" /><param name=\"movie\"
  value=\"http://www.facebook.com/v/10151026962332084\" />
  <embed src=\"http://www.facebook.com/v/10151026962332084\"
  type=\"application/x-shockwave-flash\" allowfullscreen=\"true\"
  width=\"400\" height=\"224\"></embed></object>",
"icon": "http://static.ak.fbcdn.net/rsrc.php/v2/yD/r/DggDhA4z4tO.gif",
"source": "http://video.ak.fbcdn.net/cfs-ak-snc6/v/232466/46/10151026962332084_21895.mp4?oh
  =1a6a709c2e4a3daea4eddf78a2560f79&oe=506EB8A5&__gda__
  =1349433414_8fefad34f4736161de35c25bc00d2bbe",
"created_time": "2012-10-03T13:05:29+0000",
"updated_time": "2012-10-03T13:05:29+0000",
"format": [
  {
    "embed_html": "<object width=\"130\" height=\"73\" ><param name=\"allowfullscreen\"
      value=\"true\" /><param name=\"movie\" value=\"http://www.facebook.com/v/10151026962332084\" />
      <embed src=\"http://www.facebook.com/v/10151026962332084\"
      type=\"application/x-shockwave-flash\" allowfullscreen=\"true\"
      width=\"130\" height=\"73\"></embed></object>",
    "width": 130,
    "height": 73,
    "filter": "130x130",
    "picture": "http://vthumb.ak.fbcdn.net/hvthumb-ak-snc6/
      s130x130/245287_10151026963867084_10151026962332084_24448_571_t.jpg"
  },
  {
    "embed_html": "<object width=\"480\" height=\"270\" >
      <param name=\"allowfullscreen\" value=\"true\" />
      <param name=\"movie\" value=\"http://www.facebook.com/v/10151026962332084\" />
      <embed src=\"http://www.facebook.com/v/10151026962332084\"
      type=\"application/x-shockwave-flash\" allowfullscreen=\"true\"
      width=\"480\" height=\"270\"></embed></object>",
    "width": 480,
    "height": 270,
    "filter": "480x480",
    "picture": "http://vthumb.ak.fbcdn.net/hvthumb-ak-snc6/
      s480x480/245287_10151026963867084_10151026962332084_24448_571_b.jpg"
  },
  {
    "embed_html": "<object width=\"720\" height=\"405\" >
      <param name=\"allowfullscreen\" value=\"true\" /><param name=\"movie\"
      value=\"http://www.facebook.com/v/10151026962332084\" />
      <embed src=\"http://www.facebook.com/v/10151026962332084\"
      type=\"application/x-shockwave-flash\" allowfullscreen=\"true\"
      width=\"720\" height=\"405\"></embed></object>",
    "width": 720,
    "height": 405,
    "filter": "720x720",
    "picture": "http://vthumb.ak.fbcdn.net/hvthumb-ak-snc6/
```

```
        s720x720/245287_10151026963867084_10151026962332084_24448_571_b.jpg"
      },
      {
        "embed_html": "<object width=\"1280\" height=\"720\" ><param name=\"allowfullscreen\"
         value=\"true\" /><param name=\"movie\" value=\"http://www.facebook.com/v/10151026962332084\" />
         <embed src=\"http://www.facebook.com/v/10151026962332084\"
         type=\"application/x-shockwave-flash\" allowfullscreen=\"true\"
         width=\"1280\" height=\"720\"></embed></object>",
        "width": 1280,
        "height": 720,
        "filter": "native",
        "picture": "http://vthumb.ak.fbcdn.net/hvthumb-ak-snc6/
         245287_10151026963867084_10151026962332084_24448_571_b.jpg"
      }
    ]
  }
```

从代码清单5-9中可以看到，在返回数据中包含有这段视频的详细介绍。Facebook Graph API中视频内容返回值的属性与含义可参见表5-1。视频格式（format）的主要Metadata属性及其含义可参见表5-2。

5

表5-1 Facebook Graph API中视频返回值的属性及其含义

属　性	含　义
id	视频id，此处为10151026962332084
format	视频格式，以数组形式包括3种预设尺寸的视频信息以及一种原始视频尺寸信息
updated_time	内容上载时间，此处为2012-10-03T13:05:29+0000

小技巧 基于Facebook Graph API，开发者可以获得丰富的对象信息。如果读者感兴趣，想进一步了解Facebook Graph API文档，可详见https://developers.facebook.com/docs/reference/api/。

在Graph API所返回的`format`数组中，包含有不同尺寸的Flash视频Metadata内容。在这样的Metadata中，Facebook提供了不同尺寸的视频成比例缩放之后的尺寸、缩略图位置以及嵌入脚本。

表5-2 Facebook Graph API中Flash视频的主要Metadata属性及其含义

属　性	含　义
embed_html	用于嵌入Facebook Flash视频的脚本。如果使用，需要进行HTML内容转义
width	视频宽度，单位为像素
height	视频高度，单位为像素
filter	过滤条件，如480*480表示最大宽度或最大高度为480像素，native表示为原始尺寸
picture	视频缩略图地址

基于浏览器的分辨率，开发者可以选择不同的嵌入式脚本来弹出视频窗口。例如，将之前所

返回的format数组中480*480过滤条件所对应的视频嵌入到移动应用中，相关代码如代码清单5-10所示。

代码清单5-10　嵌入Facebook视频的弹出视频代码片段

```
<div data-role="popup" id="popupVideo" data-overlay-theme="a" data-theme="c"
  style="max-width:480px;" class="ui-corner-all"><a href="#" data-rel="back" data-role="button"
  data-theme="a" data-icon="delete" data-iconpos="notext" class="ui-btn-right">Close</a>
  <object width='480' height='273' >
  <param name='allowfullscreen' value='true' />
  <param name='movie' value=' http://www.facebook.com/v/10151026962332084' />
  <embed src='http://www.facebook.com/v/10151026962332084'
    type='application/x-shockwave-flash' allowfullscreen='true' width='480' height='273'></embed>
</object>
</div>
```

此外，在处理播放器像素尺寸的scale()函数中，也需要根据Facebook Graph API所获得的长宽比例进行设置。在这个Facebook程序中，scale()函数被设置为这样：

```
var size = scale( 480, 273, 15, 1 ),
    w = size.width,
    h = size.height;
```

基于Facebook Graph API所获得的视频信息，集成到弹出视频页面的呈现效果如图5-10所示。

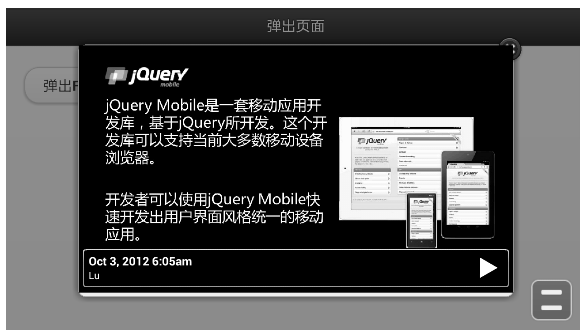


图5-10　弹出页面呈现Facebook视频

> **注意**　虽然在Facebook Graph API所获得的视频对象数据中，filter过滤条件为480*480，但是在实际设置播放器界面的时候，需要使用width和height属性，而不是过滤条件。过滤条件用以通过Graph API找到最佳分辨率的视频内容与缩略图。这是使用Facebook Graph API处理视频时需要注意的。

5.3.3　地图

将地图嵌入在移动应用中的示例图如图5-11所示。



图5-11 弹出百度地图

以集成百度地图为例，要实现弹出地图，主要有两种方法：一种方法是将地图绘制在内联框架中，然后将内联框架装入弹出页面的div容器中；另一种方法是调用百度地图接口，将地图以图片格式呈现在弹出页面中。由于后一种方法缺少交互性，我们主要介绍前一种方法。

在将百度地图通过内联方式集成在弹出页面时，首先建立一个独立的百度地图页面，这里我们将使用百度地图创建工具自动创建。开发者也可以通过百度地图API开发自己的地图页面。

小技巧 百度地图创建工具位于
http://dev.baidu.com/wiki/static/map/API/tool/creatMap/。

在百度地图创建工具中（如图5-12所示），我们在左侧输入地图创建条件，接着点击页面下方的“获取代码”按钮，获得地图页面的HTML代码，然后将这些代码单独保存在移动应用项目中。



图5-12 百度地图创建工具

在包含弹出地图页面的应用中，将生成的百度地图页面通过iframe内联框架集成在移动应用中，具体代码如下：

```
<div data-role="popup" id="popupMap" data-overlay-theme="a" data-theme="a"
  data-corners="false" data-tolerance="15,15">
  <a href="#" data-rel="back" data-role="button" data-theme="a" data-icon="delete"
    data-iconpos="notext" class="ui-btn-right">Close</a>
  <iframe src="map/baidumap.html" width="450" height="270" seamless></iframe>
</div>
```

经过上面两个步骤实现的应用在一些移动设备操作系统上运行时，有可能会出现显示异常。例如，在Android 2.3中，可能会出现因为加载iframe内联框架造成布局混乱。为此，还需要对之前的代码进行一些优化，以保证应用在不同操作系统中正常显示。

这样的优化大致要经过如下3个步骤。

(1) 添加scale()函数（关于这个函数的详细信息，请参见5.3.2节）。

(2) 添加pageinit事件响应函数。在这个函数中，设置内联框架以及内联框架页面中展现百度地图的div容器的高度和宽度。由于百度地图位于内联框架所指向的页面，所以可以使用下面的代码来设置高度：

```
$( "#popupMap iframe" ).contents().find( "#map_canvas" ).css( { "width": w, "height" : h } );
```

(3) 优化用百度地图创建工具所生成的页面。在百度地图容器之外，增加一个用作设置地图宽度和高度的div容器。在这个代码实现中，这个div容器的id为map_canvas。另外，在CSS样式定义中，设置#map_canvas的height属性高度值为100%。

使用百度地图创建工具生成的代码是不包含viewport声明的，这样可能导致在一些浏览器中显示不正常，因此需要在百度地图页面的<head>中对viewport进行如下设置：

```
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
```

调整之后，百度地图的地图容器代码参见代码清单5-11所示。

代码清单5-11　经过优化的百度地图容器代码片段

```
<div id="map_canvas">
  <!--百度地图容器-->
  <div style="width:480px;height:320px;border:#ccc solid 1px;" id="dituContent"></div>
</div>
```

经过上述优化调整后，集成在弹出页面中的百度地图就可以正常呈现在不同的浏览器中了。百度地图弹出页面的完整实现代码如代码清单5-12所示。

代码清单5-12　完整的百度地图弹出页面

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
```

```
<link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
<script src="../js/jquery-1.7.1.min.js"></script>
<script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
<script type="text/javascript">
  function scale( width, height, padding, border ) {
    var scrWidth = $( window ).width() - 30,
      scrHeight = $( window ).height() - 30,
      ifrPadding = 2 * padding,
      ifrBorder = 2 * border,
      ifrWidth = width + ifrPadding + ifrBorder,
      ifrHeight = height + ifrPadding + ifrBorder,
      h, w;

    if ( ifrWidth < scrWidth && ifrHeight < scrHeight ) {
      w = ifrWidth;
      h = ifrHeight;
    } else if ( ( ifrWidth / scrWidth ) > ( ifrHeight / scrHeight ) ) {
      w = scrWidth;
      h = ( scrWidth / ifrWidth ) * ifrHeight;
    } else {
      h = scrHeight;
      w = ( scrHeight / ifrHeight ) * ifrWidth;
    }

    return {
      'width': w - ( ifrPadding + ifrBorder ),
      'height': h - ( ifrPadding + ifrBorder )
    };
  };

  $( document ).on( "pageinit", function() {
    $( "#popupMap iframe" )
      .attr( "width", 0 )
      .attr( "height", 0 );

    $( "#popupMap iframe" ).contents().find( "#map_canvas" )
      .css( { "width" : 0, "height" : 0 } );

    $( "#popupMap" ).on({
      popupbeforeposition: function() {
        var size = scale( 480, 320, 0, 1 ),
          w = size.width,
          h = size.height;

        $( "#popupMap iframe" )
          .attr( "width", w )
          .attr( "height", h );

        $( "#popupMap iframe" ).contents().find( "#map_canvas" )
          .css( { "width": w, "height" : h } );
      },
      popupafterclose: function() {
        $( "#popupMap iframe" )
          .attr( "width", 0 )
          .attr( "height", 0 );
```

```
            $( "#popupMap iframe" ).contents().find( "#map_canvas" )
              .css( { "width": 0, "height" : 0 } );
          }
        });
      });
    </script>
  </head>
  <body>
    <section id="page1" data-role="page">
        <header data-role="header">
        <h1>弹出页面</h1>
      </header>
      <div class="content" data-role="content">
        <a href="#popupMap" data-rel="popup" data-role="button" data-inline="true">
          弹出百度地图
        </a>
        <div data-role="popup" id="popupMap" data-overlay-theme="a" data-theme="a"
          data-corners="false" data-tolerance="15,15">
          <a href="#" data-rel="back" data-role="button" data-theme="a"
            data-icon="delete" data-iconpos="notext" class="ui-btn-right">
           Close
          </a>
          <iframe src="map/baidumap.html" width="450" height="270" seamless>
          </iframe>
        </div>
      </div>
    </section>
  </body>
</html>
```

小技巧　如果开发者希望进一步了解百度地图的开发方法，可以参考下面这两个PDF文档：百度地图开发指南(详见http://dev.baidu.com/wiki/static/map/API/doc/developer.pdf)和百度地图API参考（详见http://dev.baidu.com/wiki/static/map/API/doc/classes.pdf）。

如果开发海外地图应用，可以使用Google地图。在基于jQuery Mobile开发的应用中，使用Google地图和百度地图大致上是相当的，这里我们就不再赘述了。示例图如图5-13所示。



图5-13　弹出Google地图

Google地图提供两套不同级别的地图API接口——Maps API和Maps API Premier，大多数应用使用Maps API就能满足需求。如果需要支持每天更多的API接口调用次数、更大的地图尺寸、缩放比例等特性，以及需要获得更多技术支持和保证应用支持更多应用场景，则需要用到Maps API Premier接口。

Google Maps API和Maps API Premier接口的比较请参阅 http://maps.google.com/help/maps/getmaps/compare.html。

5.3.4　覆盖面板

覆盖面板可以作为导航工具栏的扩展，当打开它时，它以半透明的样式呈现出来，如图5-14所示。在触碰面板之外的区域时，将关闭覆盖面板。



图5-14　覆盖面板

在覆盖面板中，可以包含按钮或其他表单元素。

要实现覆盖面板，大致需要下面3个步骤。

(1) 首先将各个按钮放置在弹出页面的div容器内部。下面的这段代码将按钮以mini样式放置在覆盖面板中，并设置按钮的图标样式：

```
<div data-role="popup" id="popupOverlayPanel" data-corners="false"
  data-theme="none" data-shadow="false" data-tolerance="0,0">
  <button data-theme="a" data-icon="back" data-mini="true">返回</button>
  <button data-theme="a" data-icon="grid" data-mini="true">菜单</button>
  <button data-theme="a" data-icon="plus" data-mini="true">继续</button>
</div>
```

这段代码设置了按钮的主题属性data-theme为a。如果不设置，覆盖面板将继承上一级容器的主题设置。

(2) 为了方便触控操作，可以设置触控面板中各个按钮的边距、背景颜色和宽度等。

(3) 最后，由于设置控制面板高度时，不能在CSS中使用height:100%的方法来表示，所以需要通过JavaScript将控制面板的高度设置为与浏览器屏幕的高度一致。下面的代码将高度设置绑定

在覆盖面板的popupbeforeposition事件上，在每次打开覆盖面板之前将覆盖面板的高度设置为与当前浏览器窗口的高度一样：

```
<script type="text/javascript">
  $('#popupOverlayPanel').live('popupbeforeposition', function(){
    var h = $(window).height();
  $("#popupOverlayPanel").css( "height", h );
});
</script>
```

此时我们就可以实现一个覆盖面板了。

代码清单5-13是覆盖面板的完整代码示例。

代码清单5-13 覆盖面板的完整代码

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type="text/javascript">
      $('#popupOverlayPanel').live('popupbeforeposition', function(){
        var h = $(window).height();
        $("#popupOverlayPanel").css( "height", h );
      });
    </script>
    <style>
      #popupOverlayPanel-popup {
        right: 0 !important;
        left: auto !important;
      }
      #popupOverlayPanel {
        width: 200px;
        border: 1px solid #000;
        border-right: none;
        background: rgba(0,0,0,.5);
        margin: -1px 0;
      }
      #popupOverlayPanel .ui-btn {
        margin: 2em 15px;
      }
    </style>
  </head>
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
        <h1>弹出页面</h1>
      </header>
      <div class="content" data-role="content">
        <a href="#popupOverlayPanel" data-rel="popup" data-transition="slide"
          data-position-to="window" data-role="button" data-inline="true">
```

```
            弹出覆盖面板
          </a>
          <div data-role="popup" id="popupOverlayPanel" data-corners="false"
            data-theme="none" data-shadow="false" data-tolerance="0,0">
            <button data-theme="a" data-icon="back" data-mini="true">返回</button>
            <button data-theme="a" data-icon="grid" data-mini="true">菜单</button>
            <button data-theme="a" data-icon="plus" data-mini="true">继续</button>
          </div>
        </div>
      </section>
    </body>
  </html>
```

5.4 定制弹出页面样式

在开发弹出页面的过程中，开发者可以定制弹出页面以改善用户体验，包括定制弹出页面的位置、弹出页面的关闭按钮位置、弹出动画效果以及设定主题样式等。

5.4.1 设定弹出页面的位置

设定弹出页面的位置是一个比较有用的功能。例如，设置弹出提示框的位置后，提示框会在某个特定的DOM上被打开，以实现与这个DOM相关的帮助或提示功能。

设置弹出页面位置的方法有两种：一种是在激活弹出页面的超级链接按钮中设置data-position-to属性；另一种是通过JavaScript方法对弹出页面执行open()操作，并在open()方法中设置打开弹出页面的坐标位置，这将在5.5节中详细介绍。

data-position-to属性有3个取值，详情可参见表5-3。

表5-3 data-position-to属性及其含义

属 性 值	含 义
window	弹出页面在浏览器窗口中间弹出
original	弹出页面在当前触发位置弹出
#DomId	弹出页面在DOM对象所在位置被弹出。此处需要将DOM对象id赋值给data-position-to属性，例如data-position-to="#Penguins"

代码清单5-14分别呈现3种不同的提示框位置。

代码清单5-14 弹出页面位置

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
```

```
    </head>
    <body>
      <section id="page1" data-role="page">
        <header data-role="header">
          <h1>弹出页面</h1>
        </header>
        <div class="content" data-role="content">
          <a href="#popupPenguins" data-rel="popup" data-role="button"
            data-position-to="#Penguins" data-inline="true">图片上</a>
          <a href="#popupWindow" data-rel="popup" data-role="button"
            data-position-to="window"  data-inline="true">屏幕中间</a>
          <a href="#popupOrigin" data-rel="popup" data-role="button"
            data-position-to="origin" data-inline="true">当前位置</a>
          <div style="height:40px;"></div>
          <img src="images/Penguins.jpg" width="350px" style="max-width:200px" alt="企鹅"
            id="Penguins"/>
          <div data-role="popup" id="popupPenguins">
            <p>位于图片上方的提示框。
            <p>
          </div>
          <div data-role="popup" id="popupWindow">
            <p>位于屏幕中央的提示框。
            <p>
          </div>
          <div data-role="popup" id="popupOrigin">
            <p>位于当前位置的提示框。
            <p>
          </div>
        </div>
      </section>
    </body>
  </html>
```

运行上述代码，所得的提示框效果如图5-15所示。



图5-15　处于不同位置的弹出页面

5.4.2 动画切换效果

在呈现过程中，有10种动画弹出效果供开发者选择。当需要以动画效果呈现弹出页面时，可以在打开页面的超级链接按钮中设置data-transition属性为相应动画效果即可。例如，设置某个弹出页面以中央弹出动画方式呈现的代码如下：

```
<a href="#popupSomeDialog" data-rel="popup" data-transition="pop" data-role="button">
  弹出页面
</a>
```

主要的动画方式及其对应的data-transition属性如表5-4所示。

表5-4 弹出页面动画效果

动画方式	data-transition属性值
横向幻灯方式	slide
自上向下幻灯方式	slideup
自下向上幻灯方式	slidedown
中央弹出	pop
淡入淡出	fade
旋转弹出	flip
横向翻转	turn
缩小并以幻灯方式切换	flow
淡出方式显示，横向幻灯方式退出	slidefade
无动画效果	none

5

注意 并非所有动画效果都可以被移动设备所支持。例如，在早期的Android操作系统中，3D的页面切换效果是不支持的，此时会以淡入淡出效果呈现弹出动画效果。

5.4.3 弹出页面主题

要设置弹出页面主题，可以使用theme和data-overlay-theme这两种属性，其中前者用于设置弹出页面自身的主题和色板配色，后者主要用于设置弹出页面周边的背景颜色。

实现页面主题设定的代码片段如下：

```
<div data-role="popup" id="popupDialog" data-overlay-theme="a" data-theme="c">
  弹出页面内容
</div>
```

如果不设置theme属性，弹出页面将继承上一级DOM容器的主题和色板设定。例如，页面的theme设置为a，那么如果不特别进行theme主题设定，则其下的各个弹出页面都将继承theme为a的设置。

有别于theme属性继承自上一级DOM容器的主题设定，如果data-overlay-theme没有设置，那么呈现弹出页面的时候，弹出页面的周边是没有颜色覆盖的，示例图如图5-16左图所示。如果data-overlay-theme属性设置为a，则呈现效果如图5-16右图所示。



图5-16　弹出页面主题data-overlay-theme设置效果

5.4.4　关闭按钮

在之前的弹出图片中，右上角有一个关闭按钮，点击这个按钮就可以关闭弹出图片，如图5-17所示。



图5-17　弹出图片

要实现关闭按钮，可以在div容器开始的位置添加一个超级链接按钮。在这个超级链接按钮中，设置data-rel属性为back，即点击这个按钮相当于返回上一页。如果希望图标位于右上角，则设置这个超级链接按钮的class属性为ui-btn-right，如果希望按钮出现在左上角，则设置该属性为ui-btn-left。

最后，可以设置按钮的文字和图标。如果希望只显示一个图标按钮而不包含任何文字，则设置data-iconpos属性为notext。代码清单5-15演示了关闭按钮的代码。

代码清单5-15　关闭按钮

```
<div data-role="popup" id="popupPhoto">
  <!--关闭按钮-->
  <a href="#" data-rel="back" data-role="button" data-theme="a" data-icon="delete"
    data-iconpos="notext" class="ui-btn-right">Close</a>
  <!--这里是弹出图片内容-->
</div>
```


5.5　属性、选项、方法和事件

在开发移动应用的时候，可以通过设置属性、选项、方法以及绑定事件响应函数来开发弹出页面的功能。

5.5.1　属性

属性定义在弹出页面的DOM对象上，用以设定弹出页面的样式、主题等内容，如表5-5所示。

表5-5　属性

属　　性	含　　义
data-corners	设置弹出页面外形为直角或者圆角。默认为true，圆角外形。 示例代码：<div data-role="popup"id="popupId"data-corners="false">…</div>
data-overlay-theme	设置弹出页面周边的色板。色板的设定将影响周围区域的背景颜色，如果不设置，则弹出页面周围没有背景色。 示例代码：<div data-role="popup" id="popupId"data-overlay-theme="a">…</div>
data-position-to	设置弹出页面的位置。 默认为origin，表示触发的当前位置。 如果为window，则表示位于浏览器中央。 如果设置为某个DOM对象id，则弹出页面将会呈现在这个DOM对象上方。 示例代码：<div data-role="popup" id="popupId"data-position-to="window">…</div>
data-shadow	设置弹出页面周边是否有阴影效果。 默认为true，有阴影效果。如果为false，则没有阴影效果。 示例代码：<div data-role="popup" id="popupId"data-shadow="true">…</div>

（续）

属性	含义
data-theme	设置弹出页面主题样式。默认为空，即主题样式继承自上一层容器。 示例代码：<div data-role="popup" id="popupId"data -theme="a">…</div>
data-transition	设置弹出页面的动画效果。默认为none，没有动画效果。 这个属性不是设置在弹出页面的div容器上，而是设置在打开弹出页面的超级链接按钮上。 示例代码：…

注意 除了弹出页面动画切换效果是在超级链接按钮之外设定的，其他与弹出页面相关的属性都是在div容器上设置的。

5.5.2 选项

很多弹出页面选项与属性的实现效果是类似的。选项通常是通过JavaScript对所有弹出页面进行设置，或者通过特定筛选器对筛选出的弹出页面进行设置，具体如表5-6所示。

表5-6 弹出页面选项与属性对照表

选项	属性	功能
corners	data-corners	设置弹出窗口外形为直角或者圆角
overlayTheme	data-overlay-theme	设置弹出页面周边的色板，其设定将影响周围区域的背景颜色
positionTo	data-position-to	设置弹出页面的位置
shadow	data-shadow	设置弹出窗口以阴影方式显示
theme	data-theme	设置弹出窗口的主题样式
transition	data-transition	设置弹出页面的动画效果

在弹出页面中，还有一些选项没有对应的属性定义，这些选项在开发过程中经常用到，具体如表5-7所示。

表5-7 弹出页面中没有对应属性的选项列表

选项	含义
initSelector	用于自定义弹出页面的CSS选择器。设置initSelector之后，所设置的DOM将被呈现为弹出页面
tolerance	用于设置弹出页面距离浏览器边界的最小尺寸。 默认为30,15,30,15。 如果不设置，则表示使用默认值。 如果设置1个值，则表示四边的边距都采用设置值。 如果设置2个值，则表示第一个值用于上边距和下边距，第二个值用于左边距和右边距。 如果设置4个值，则表示第一个值用于上边距，第二个值用于右边距，第三个值用于下边距，第四个值用于左边距

5.5.3　方法

对于弹出页面而言，可以通过JavaScript语句操作open()和close()这两个方法来打开和关闭。在打开弹出页面时，可以设置如表5-8所示的这些选项。

表5-8　弹出页面打开选项

打开选项	含　义
X	打开弹出页面的*X*坐标
Y	打开弹出页面的*Y*坐标
transition	打开弹出页面的动画效果
positionTo	打开弹出页面的位置

如果没有设置弹出页面中的坐标位置或positionTo弹出页面位置属性，则默认会在当前浏览器窗口的中央打开弹出页面。这与通过超级链接按钮打开的弹出页面不同，通过超级链接按钮打开的弹出页面默认位于超级链接按钮的上方。

5.5.4　事件

打开和关闭弹出页面时，将会触发弹出页面事件，具体如表5-9所示。

表5-9　弹出页面事件

事　件	含　义
popupbeforeposition	在弹出页面已经被处理完成而准备呈现之前，将会触发此事件。 对于视频、图片、地图等的大多尺寸设置操作都在这个事件中完成
popupafteropen	弹出页面完全呈现完成时，将会触发此事件
popupafterclose	弹出页面完全关闭时，将会触发此事件

第 6 章

触控交互

6

在将触屏应用在手机之前，每部手机至少拥有一个数字键盘或者QWERTY键盘用以输入电话号码或者文字。而在现在的大多移动设备上，我们可以直接用手在触摸屏上“书写”，通过在触摸屏上点击或者在虚拟键盘上输入，或者基于手势向移动设备发出指令，而移动设备会响应这些操作指令。jQuery Mobile支持触控交互，基于此，开发者可以开发出不同的触控交互应用。

在这一章中，我们将会了解到：

- 轻击（tap）或者按住（taphold）操作；
- 轻扫屏幕操作；
- 虚拟鼠标事件。

此外，还有一些其他的触控操作，例如基于虚拟键盘执行输入操作，但这些操作是基于移动设备操作系统或输入法实现的，超出了jQuery Mobile的讨论范围，这里就不介绍了。

6.1 触控事件

当使用移动设备进行触控操作时，最常用的就是轻击、按住屏幕或者手势操作，jQuery Mobile可以通过绑定的触控事件来响应使用者的特定触控行为。

6.1.1 轻击与按住

轻击操作就是快速触碰触摸屏。当触碰操作结束时，会触发tap事件。如果按住触摸屏不放，就会触发taphold事件。虽然tap被称为轻击，但是其实与施加在触摸屏上的力是没有关系的，只是“碰”了一下屏幕。

下面的代码展现了绑定与执行tap事件和taphold事件的方法：

```
$('#page1').live('tap', function(){
  $.mobile.changePage('#page2');
});

$('#page1').live('taphold',function(){
  alert('taphold事件被触发');
});
```

一般情况下，tap或taphold被绑定在某个DOM容器上。在这个程序中，tap和taphold事件被绑定在页面上。当触发tap事件时，页面跳转到另一个页面。当触发taphold事件时，则会发出一个消息对话框，提示“taphold事件被触发”。

轻击与按住的完整代码示例如代码清单6-1所示。

代码清单6-1　轻击与按住事件的代码示例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type='text/javascript'>
      $('#page1').live('tap', function(){
        $.mobile.changePage('#page2');
      });

      $('#page2').live('tap', function(){
        $.mobile.changePage('#page1');
      });

      $('#page1').live('taphold',function(){
        alert('taphold事件被触发');
      });

      $('#page2').live('taphold',function(){
        $.mobile.changePage('#about');
      });
    </script>
  </head>
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
        <h1>Tap事件处理</h1>
      </header>
      <div class="content" data-role="content">
        轻击页面进入下一页。<br/>
        按住不放，打开关于对话框
      </div>
      <footer data-role="footer">
      </footer>
    </section>
    <section id="page2" data-role="page">
      <header data-role="header">
        <h1>Tap事件处理</h1>
      </header>
      <div class="content" data-role="content">
        轻击页面返回前一页。
```

```
      </div>
      <footer data-role="footer">
      </footer>
    </section>
    <div id='about' data-role='dialog'>
      <div data-role="header">
        <h1>关于本程序</h1>
      </div>
      <div data-role='content'>
        演示轻击触控事件响应
      </div>
    </div>
  </body>
</html>
```

在不同的移动设备浏览器中，page1和page2所触发的taphold事件对应的处理程序是不同的：page1的taphold触发一个alert对话框（如图6-1左图所示），page2将触发一个关于对话框（如图6-1右图所示），这两种对话框的执行效果也是不同的。



图6-1　通过taphold事件触发消息框或对话框

在Android 4.0.3中，在page1中触发taphold事件后，会激活一个alert消息框。如果用户需要关闭这个消息框，则要再次点击OK按钮。在page2中触发taphold事件后，会打开一个关于对话框。当用户的手从屏幕上离开后，就会自动关闭关于对话框。而这个关闭操作有别于之前page1中的消息框，不需要通过点击按钮就可以实现。

在不同的浏览器中执行taphold事件触发程序，对话框的执行效果也可能会有所不同。有别于之前Android 4.0.3虚拟设备中的执行效果，在Chrome、Internet Explorer和Firefox中，当释放page2的鼠标之后，对话框并不会自动消失，使用者需要点击关闭按钮才能关闭这个对话框。

6.1.2　轻扫

在jQuery Mobile中，轻扫操作（swipe）是用手指或手写笔快速在触摸屏上向左或向右划过。在用户以轻扫方式操作程序时，会触发swipeleft事件或者swiperight事件。

代码清单6-2演示了将轻扫事件与页面绑定的示例代码。向左轻扫时，则触发对于DOM对象id为lnkDialog的超级链接的点击操作，向右轻扫时则更新页面到另一个页面。

代码清单6-2　轻扫事件示例代码

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type='text/javascript'>
      $('#page1').live('swiperight', function(){
        $.mobile.changePage('#page2');
      });

      $('#page2').live('swiperight', function(){
        $.mobile.changePage('#page1');
      });

      $('#page1').live('swipeleft',function(){
        $('#lnkDialog').click();
      });

      $('#page2').live('swipeleft',function(){
        $('#lnkDialog').click();
      });
    </script>
  </head>
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
        <h1>swipe事件处理</h1>
      </header>
      <div class="content" data-role="content">
        向右滑动页面进入下一页。<br/>
        向左滑动页面，打开关于对话框
      </div>
      <footer data-role="footer">
      </footer>
    </section>
    <section id="page2" data-role="page">
      <header data-role="header">
```

```
      <h1>swipe事件处理</h1>
    </header>
    <div class="content" data-role="content">
      向右滑动页面进入前一页。<br/>
      向左滑动页面，打开关于对话框
    </div>
    <footer data-role="footer">
    </footer>
  </section>
  <div id='about' data-role='page'>
    <div data-role="header">
      <h1>关于本程序</h1>
    </div>
    <div data-role='content' data-theme='c'>
      演示swipeleft&swiperight触控事件响应
    </div>
  </div>
  <a id='lnkDialog' href="#about" data-rel="dialog" data-transition="pop"
    style='display:none;'></a>
 </body>
</html>
```

在打开这段代码的第一个页面时，向右轻扫则可以进入下一个页面，而向左轻扫则会打开关于对话框。

在触控程序中，如果打开下一个页面，则可以通过changePage()方法实现。例如，下面的代码将可以使页面跳转到#page1：

```
$.mobile.changePage('#page1');
```

如果希望通过触控操作打开一个指向对话框的链接，并且这个打开操作中还能定义特定的页面切换特效，那就需要一些额外的技巧。因为jQuery Mobile对话框不能使用changePage()函数打开，而必须使用超级链接以Ajax的方式打开，所以需要使用JavaScript模拟点击这个超级链接。

首先，建立一个不可见的超级链接，设置这个超级链接以对话框的方式打开页面。设置超级链接不可见，可以通过设置style属性为display:none;来实现。例如，下面的超级链接是一个不可见的链接，点击之后将以pop方式打开：

```
<a id='lnkDialog' href="#about" data-rel="dialog" data-transition="pop" style='display:none;'></a>
```

然后，在触控操作所绑定的代码中通过JavaScript实现对这个不可见超级链接的点击，就可以打开对话框了：

```
$('#page1').live('swipeleft',function(){
  $('#lnkDialog').click();
});
```

6.2　虚拟鼠标事件

虚拟鼠标事件用以对移动设备触控行为进行响应。有别于前面所提到的触控事件，虚拟鼠标

事件能够提供更具体的有关触控的操作行为信息，例如具体的触控行为以及不同坐标系下的触控坐标位置，因此，开发者可以据此定制不同的触控事件响应程序。主要的虚拟鼠标事件如表6-1所示。

表6-1　虚拟鼠标事件

事　件	含　义
vmouseover	在触控或者鼠标滑动于DOM容器之上时触发
vmouseout	在触控或者鼠标滑动离开DOM容器时触发
vmousedown	开始触摸或者鼠标被按下时触发
vmousemove	触摸滑动或鼠标移动时触发
vmouseup	触摸结束或鼠标按键被释放时触发
vclick	触摸结束或鼠标按键被释放时触发。 vclick事件通常在vmouseup事件后的300ms触发
vmousecancel	触控事件中发起mousecancel事件时触发。 例如，非常快速地执行轻扫操作时

虚拟鼠标事件与触控事件有所不同：触控事件通常并不会返回用户操作的位置，虚拟鼠标事件在触发的时候，不但可以获得当前使用者的行为，还可以获得触发虚拟鼠标事件的坐标位置。根据所处位置的不同，可以获得当前所在页面、当前屏幕以及当前窗口区域的坐标位置。虚拟鼠标坐标的定义如表6-2所示。

表6-2　虚拟鼠标坐标的定义

坐　标	含　义
pageX	当前HTML页面横坐标
pageY	当前HTML页面纵坐标
screenX	当前屏幕横坐标
screenY	当前屏幕纵坐标
clientX	当前窗口区域的横坐标
clientY	当前窗口区域的纵坐标

代码清单6-3展示了触控操作获得不同坐标体系下的坐标的示例代码。为了能够更加清晰地表现坐标体系的差异，我们需要展示页面发生滚屏之后的坐标变化。所以，在代码清单6-3的示例程序中，我们加入了500个像素高度的`<div>`，并在最后一屏的底部出现“内容底部”字样。

通常，在页面开始的一屏中，各个坐标体系的数值是一样的，但是随着输出内容的增加，一旦出现滚屏，移动设备浏览器的*Y*坐标值就会不同。有趣的是，通常移动设备浏览器都会全屏呈现，所以*X*坐标的数值一般来说在各个坐标体系下都是一样的。

代码清单6-3　触控操作获得不同坐标体系下坐标的代码

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type='text/javascript'>
      $('#page1').live('vmouseup', function(event, ui){
        alert("当前点击位置"+
          "\n"+
          "\npageX: "+event.pageX+
          '\npageY: '+event.pageY+
          "\n"+
          '\nscreenX: '+event.screenX+
          '\nscreenY:'+event.screenY+
          "\n"+
          '\nclientX: '+event.clientX+
          '\nclientY: '+event.clientY);
      });
    </script>
  </head>
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
        <h1>vMouse事件处理</h1>
      </header>
      <div class="content" data-role="content">
        轻击页面，显示点击位置。
      </div>
      <div style="height:500px;"></div>
      内容底部
      <footer data-role="footer">
      </footer>
    </section>
  </body>
</html>
```

运行上述代码，得到的运行结果如图6-2所示。

注意　仔细比较会发现，代码清单6-3中的字符\n并没有在图6-2中显示出来。在JavaScript中，\n是一个转义字符，表示为换行，所以所有标记为\n之后的文字都显示在新的一行。而连续两个\n则显示为两句话中间多了一个空行。

从图6-2中可以看出，不同坐标体系的尺寸是不同的。在移动设备中，*X*坐标的位置通常都是固定的，这也是由于移动设备中浏览器是以全屏方式显示的。如果不是全屏方式，那么*X*坐标也

会不同。同样，对于移动设备浏览器，由于浏览器是满屏显示的，所以获得的屏幕坐标和窗口区域坐标是一致的。



图6-2　第一屏点击的坐标位置与最后一屏的鼠标位置

当页面尺寸超过移动设备浏览器一屏的尺寸时，随着页面滚动鼠标的Y坐标在不同坐标体系中会变得不同。在图6-1中，当屏幕被拖动到最后一屏后，pageY的坐标与screenY和clientY的坐标是不同的，前者为221，而后面两个均为117。这是由于pageY坐标是根据页面的位置计算来的，而其他两个则是根据移动设备浏览器屏幕坐标或当前窗口区域坐标来计算的。

第 7 章

按　　钮

按钮是用户界面交互中最常用到的组件。通过点击按钮，可以提交内容，或者展开下一个对话框。由于Web移动应用页面是通过网页的形式展现的，所以这里接触到的按钮除了具有传统网页开发按钮的功能外，一些超级链接也会以按钮的样式呈现，而这正是jQuery Mobile按钮的特别之处。

在这一章中，我们将会了解到：

- 内联按钮；
- 按钮图标；
- Mini按钮；
- 按钮组；
- 开发按钮功能的属性、选项、方法和事件；
- 自定义按钮。

7.1　基本概念

在Web移动应用中，按钮有两种常见的形式：一种是表单按钮，例如实现提交、清除等功能的按钮。另一种是将超级链接美化成按钮的样式，点击后页面会跳转到新的一页。

按钮美化大体遵循如下两个原则。

- 默认的按钮类型或图片按钮，例如`input type="image"`，会自动转化为按钮样式。
- 对于超级链接添加按钮属性`data-role="button"`，就可以将超级链接美化为按钮的样式。

表7-1列出了会被美化的各种按钮。

表7-1　按钮标签定义

类　　别	按钮代码
超级链接	`<a href="#" data-role="button">超级链接按钮</a>`
按钮	`<button>表单按钮</button>`
提交按钮	`<input type="submit">提交按钮</input>`
重置按钮	`<input type="reset">重置按钮</input>`

代码清单7-1展示了超级链接按钮和表单按钮的不同实现。在这个示例中，显示“关于”字样的按钮是一个超级链接按钮，这里通过声明data-role属性为button来美化超级链接样式按钮：

```
<a href="#about" data-role="button">关于</a>
```

代码清单7-1　表单按钮与超级链接按钮的实现示例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type="text/javascript">
      function OnBtnClick()
      {
        alert('按钮被点击');
      }
    </script>
  </head>
  <body>
    <section id="page_LayoutButton" data-role="page" data-title="内置按钮样式">
      <header data-role="header">
        <h1>内置按钮样式</h1>
      </header>
      <div class="content" data-role="content" >
        <button onClick="OnBtnClick();">表单按钮</button>
        <a href="#about" data-role="button">关于</a>
      </div>
    </section>
    <div id="about" data-role="dialog">
      <div data-role="header">
        <h1>关于我们</h1>
      </div>
      <div data-role="content">演示不同按钮功能</div>
    </div>
  </body>
</html>
```

运行上述代码，得到的结果如图7-1所示。



图7-1　链接按钮和表单按钮样式

从图7-1中可以看到，表单按钮和超级链接按钮的呈现效果完全相同。

7.2　内联按钮

默认情况下，每个按钮几乎占满整个移动设备屏幕的宽度，一行只能容纳一个按钮，这在屏幕较小的移动设备下便于触控操作，而在屏幕较大的移动设备中，按钮尺寸就显得过大了，而内联按钮能有效改善这样的用户体验。

通常，内联按钮的宽度受按钮文字的数量影响。如果按钮中的文字较少，那么内联按钮的宽度也会相应变窄。这样多个内联按钮就可以方便地并列排在一起，而不是自上而下依次排列。

在按钮标签内部将data-inline属性设置为true，则按钮宽度将适应按钮文字宽度而缩小。而按钮排列顺序也不再只是按照自上而下的顺序排列，在一行能够容纳多个按钮的情况下按钮将按照自左向右的顺序依次排列。

内联按钮的代码片段如下，呈现效果如图7-2所示：

```
<button data-inline="true" onClick="OnBtnClick();">表单按钮</button>
<a href="#about" data-role="button" data-inline="true">关于</a>
```



图7-2 内联按钮

7.3 按钮图标

jQuery Mobile定义了一个data-icon属性，基于这个属性，程序可以呈现标准化的按钮图标。这是按钮图标的代码片段：

```
<button data-icon="arrow-l">向左</button>
```

所呈现的按钮样式如图7-3所示。



图7-3 向左按钮

7.3.1 按钮图标样式

data-icon可以指定不同的属性值，不同的属性值所呈现的图标也不同，具体如表7-2所示。

表7-2 data-icon的值及其含义

属 性 值	含 义
arrow-l	向左按钮。示例代码： <button data-icon="arrow-l">向左</button> 效果为： 向左
arrow-r	向右按钮。示例代码： <button data-icon="arrow-r">向右</button>

（续）

属性值	含义
arrow-r	效果为：
arrow-u	向上按钮。示例代码： <button data-icon="arrow-u">向上</button> 效果为：
arrow-d	向下按钮。示例代码： <button data-icon="arrow-d">向下</button> 效果为：
delete	删除按钮。示例代码： <button data-icon="delete">删除</button> 效果为：
plus	加号按钮。示例代码： <button data-icon="plus">加号</button> 效果为：
minus	减号按钮。示例代码： <button data-icon="minus">减号</button> 效果为：
check	检查按钮。示例代码： <button data-icon="check">检查</button> 效果为：
gear	齿轮按钮。示例代码： <button data-icon="gear">齿轮</button> 效果为：
refresh	刷新按钮。示例代码： <button data-icon="refresh">刷新</button> 效果为：
forward	向前按钮。示例代码： <button data-icon="forward">向前</button> 效果为：
back	向后按钮。示例代码： <button data-icon="back">向后</button> 效果为：

（续）

属　性　值	含　　义
grid	网格按钮。示例代码： <button data-icon="grid">网格</button> 效果为：
star	星形按钮。示例代码： <button data-icon="star">星形</button> 效果为：
alert	警告按钮。示例代码： <button data-icon="alert">警告</button> 效果为：
info	信息按钮。示例代码： <button data-icon="info">信息</button> 效果为：
home	主页按钮。示例代码： <button data-icon="home">主页</button> 效果为：
search	查找按钮。示例代码： <button data-icon="search">查找</button> 效果为：

图7-4是在Android虚拟设备中所呈现的按钮图标样式。



图7-4　按钮图标

7.3.2　按钮图标位置

通过data-iconpos属性，可以设置图标在按钮中的位置，具体如表7-3所示。

表7-3　按钮图标位置

图标位置	含　义
left	图标位于左侧（默认值）。通常不用设置，因为默认位置就是屏幕左侧
right	图标位于右侧
top	图标位于上方正中
bottom	图标位于下方正中
notext	只显示图标而不显示按钮文字

代码清单7-2演示了不同的按钮图标位置的代码。

代码清单7-2　按钮图标位置代码片段

```
<div class="content" data-role="content">
  <button data-icon="home" data-iconpos="left">主页</button>
  <button data-icon="home" data-iconpos="right">主页</button>
  <button data-icon="home" data-iconpos="top">主页</button>
  <button data-icon="home" data-iconpos="bottom">主页</button>
  <div style="height:15px;"></div>
  <hr>
  notext类型按钮:
  <button data-icon="home" data-iconpos="notext">主页</button>
</div>
```

在Android虚拟设备上运行上述代码，得到的效果如图7-5所示。



图7-5　按钮图标位置

如果只设置图标，而不希望包括任何文字，则可以设置属性data-iconpos为notext。在图7-5中，最后一个按钮就是这种样式。

7.4 mini 按钮

在一些场景中（例如按钮和其他表单组件被放置在折叠内容块中），由于移动设备自身的屏幕尺寸限制，加之折叠内容块的显示区域比移动设备浏览器的显示区域小，如果使用标准尺寸的按钮和表单组件，那么内容布局和呈现将拥挤而不便操作，为此可以通过设置data-mini属性为true的方式，将按钮或者表单组件以mini方式呈现，如图7-6所示。



图7-6　左图为mini按钮，右图为正常尺寸按钮

示例代码如下：

```
<button data-mini="true" data-icon="home">主页</button>
```

当然，其他表单元素也可以设置为mini尺寸，比如文本框和复选框。

从jQuery Mobile 1.2.0开始，工具栏中的按钮将默认以mini方式显示。而在之前的jQuery Mobile 1.1.1或更早的版本中，默认的尺寸是正常尺寸，除非开发者特别设置按钮为mini方式。在涉及将jQuery Mobile早期程序迁移到jQuery Mobile 1.2.0或之后版本的时候，开发者需要注意这个变化。

7.5 按钮组

一组相关的按钮可以通过按钮组的方式分组排列在一起，经过分组之后的按钮可以横向排列或纵向排列。

实现按钮组的效果很简单：在div容器中，将data-role属性设置为controlgroup，即可将一组按钮以纵向方式排列在一起。如果希望以横向方式排列，只需设置data-type属性为horizontal即可。

纵向按钮组的示例代码如下：

```
<div data-role="controlgroup">
  <a href="#"  data-role="button" >链接按钮</a>
  <button >表单按钮</button>
</div>
```

横向按钮组的示例代码如下：

```
<div data-role="controlgroup" data-type="horizontal">
  <a href="#"  data-role="button" >链接按钮</a>
  <button >表单按钮</button>
</div>
```

如果需要将按钮组设置为mini样式，则需要在按钮组容器中添加data-mini="true"属性：

```
<div data-role="controlgroup" data-type="horizontal" data-mini="true">
```

上述代码的运行效果如图7-7所示。



图7-7　左侧为标准尺寸按钮组，右侧为mini按钮组

在按钮组中，不但可以使用文字按钮，同样也可以在按钮中加入图标，或者使用纯粹的图标按钮。

在jQuery Mobile 1.1.1或者之前版本的移动Web应用中，不建议使用无文字的图标按钮。因为无文字的图标按钮尺寸非常小，不方便用户的触控操作。这个问题在jQuery Mobile 1.2.0之后的版本得到改善。如果你使用的是jQuery Mobile最新版本，建议根据自己的应用场景做一下测试。代码清单7-3演示了无文字的图标按钮组示例代码。

代码清单7-3 无文字的图标按钮组

```
<div class="content" data-role="content">
  按钮组图标:
  <div style="height:10px"></div>
    <div data-role="controlgroup" data-type="horizontal" >
      <a href="index.html" data-role="button" data-icon="arrow-u" data-iconpos="notext">
        Up
      </a>
      <a href="index.html" data-role="button" data-icon="arrow-d" data-iconpos="notext">
        Down
      </a>
      <a href="index.html" data-role="button" data-icon="delete" data-iconpos="notext">
        Delete
      </a>
    </div>
  </div>
</div>
```

在jQuery Mobile 1.2.0及后续版本环境下执行代码清单7-3，得到的运行结果如图7-8所示。

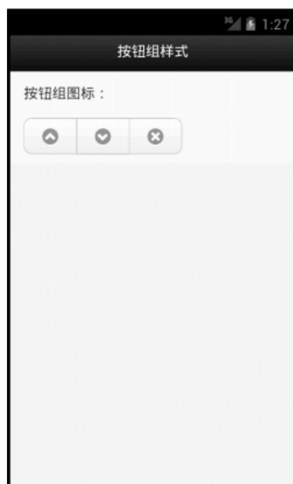


图7-8 图标按钮组

7.6 按钮属性、选项、方法与事件

在写HTML页面的时候，开发者可以通过设定属性控制按钮的呈现样式，也可以通过JavaScript根据上下文环境对按钮进行样式控制和事件响应。下面我们就从属性、选项、方法和事件这4个方面来介绍一下按钮的设定。

7.6.1 属性

按钮属性是定义在按钮容器内部的属性，用于确定所使用的按钮样式，如直角按钮或者圆角

按钮、按钮图标样式、按钮图标位置、按钮配色风格等，具体如表7-4所示。

表7-4　按钮属性

属　　性	含　　义
data-corners	设置按钮外形为直角或者圆角，默认为true，表示圆角外形。 示例代码： `<button data-corners="false">按钮</button>`
data-icon	设置按钮图标样式，默认为null，表示不显示图标。 示例代码： `<button data-icon="home">按钮</button>`
data-iconpos	设置图标按钮位置，默认为left，表示图标位于按钮左侧。 示例代码： `<button data-iconpos="right" data-icon="home">按钮</button>`
data-iconshadow	设置图标按钮是否呈现阴影效果，默认为true，表示显示阴影效果。 示例代码： `<button data-mini="true" data-icon="home">按钮</button>`
data-inline	设置按钮是否为内联按钮，默认为null，表示不启用内联按钮样式。 示例代码： `<button data-inline="true">按钮</button>`
data-mini	设置按钮是否为mini尺寸，默认为false，表示正常尺寸显示。 示例代码： `<button data-mini="true">按钮</button>`
data-shadow	设置按钮为阴影方式显示，默认为true，表示显示按钮外侧阴影。 示例代码： `<button data-shadow="false">按钮</button>`
data-theme	设置按钮显示theme风格，默认为null，表示继承上层theme风格。 示例代码： `<button data-theme="c">按钮</button>`

7.6.2　选项

大部分按钮选项与属性所实现的效果是非常类似的，表7-5列出了类似功能的选项与属性的对照表。通常，按钮选项通过jQuery筛选器进行选择，并将效果批量施加在特定DOM对象上。

表7-5　按钮选项与属性对照表

选　　项	属　　性	功　　能
corners	data-corners	设置按钮外形为直角或者圆角
icon	data-icon	设置按钮图标样式

（续）

选　　项	属　　性	功　　能
iconpos	data-iconpos	设置图标按钮位置
iconshadow	data-iconshadow	设置图标按钮是否呈现阴影效果
inline	data-inline	设置按钮是否为内联按钮
mini	data-mini	设置按钮是否为mini尺寸
shadow	data-shadow	设置按钮为阴影方式显示
theme	data-theme	设置按钮显示theme风格

在jQuery Mobile中，initSelector选项是唯一没有按钮属性对应的选项，它用于美化特定CSS选择器指定的按钮。jQuery Mobile会将initSelector中的选择器所指向的DOM美化成按钮的样子。

initSelector的默认值是5种按钮对象button，[type='button']，[type='submit']，[type='reset']，[type='image']。也就是说，在没有任何initSelector设定的情况下，jQuery Mobile会默认将这5种按钮美化为jQuery Mobile的按钮风格。

此外，开发者还可以设定特定的jQuery选择器，例如特定的CSS，以实现对特定DOM对象的美化。

7

7.6.3 方法和事件

对于Form类型的按钮，可以通过按钮方法对其执行启用、禁用或刷新操作，具体如表7-6所示。

表7-6 按钮方法

方　　法	功　　能
enable	启用一个被禁用的按钮
disable	禁用一个Form按钮
refresh	刷新按钮，用于更新按钮显示样式

代码清单7-4禁用了id为btnSearch的查找按钮。

代码清单7-4 禁用按钮的代码示例

```
<script type="text/javascript">
  function OnBtnClick(){
    $("#btnSearch").button('disable');
  }
</script>
```

运行上述代码，可以发现查找按钮被禁用后的效果如图7-9所示。



图7-9　按钮被禁用的效果

注意　在jQuery Mobile中，按钮方法只能用于表单按钮，而不可用于按钮样式的超级链接。

需要说明的是，在创建按钮时，将触发create事件。

7.7　自定义按钮

jQuery Mobile提供了常用的按钮样式和18种按钮图标，但在一些特定场景中，用户需要根据自己的需要定制按钮，例如信息系统中的报表和打印图标按钮，或者需要在按钮中放入大量文字时。

7.7.1　自定义按钮图标

除了可以使用jQuery Mobile提供的18种按钮图标之外，我们还可以自定义按钮图标，下面我们详细介绍一下如何自定义按钮图标。

首先，开发者绘制出PNG-8格式的18×18像素的图标文件。这里开发者可以使用透明背景的格式，这样有利于与页面和按钮主题风格集成。

之后，开发者需要在自定义的CSS文件中添加特定的图标样式。这里需要说明的是，按钮图标样式命名需要以.ui-icon-开头。比如，一个自定义报表图标按钮可以被命名为.ui-icon-report，这样便于jQuery Mobile程序识别它并将其作为按钮图标加载。

下面是一段自定义报表图标的CSS样式：

```
<style>
  .ui-icon-report {
    background-image: url(report.png);
    background-size: 18px 18px;
  }
</style>
```

需要说明的是，实现自定义按钮图标时，需要将样式名称的.ui-icon-前缀部分去掉，将名称剩下的部分report赋值给data-icon属性：

```
<button data-icon="report">报表按钮</button>
```

如果用户使用的是iPhone 4、iPhone 4S或者The New iPad，由于这些设备采用高分辨屏幕，所以开发者最好能够开发一套用于高分辨率的按钮，以保证用户体验良好。一般情况下，按钮的分辨率为36×36像素。

对不同移动设备浏览器的分辨率而加载不同的CSS定义，这属于CSS3的媒体查询技术，相关内容请参阅第11章。

```
@media only screen and (-webkit-min-device-pixel-ratio: 2) {
  .ui-icon-report {
    background-image: url("report.HD.png");
    background-size: 18px 18px;
  }
}
```

如果浏览器检测到当前的显示设备的分辨率比较高，就会应用媒体查询中所指向的按钮图标样式。

7.7.2 文字折行显示

在一些应用场景下，我们可能需要在按钮上放置长度不等的文字，此时就支持文字折行显示。例如，在国际化和本地化过程中，就可能遇到不同语言转换中文字长度不等的问题，如在将中文翻译为英文的过程中，文字通常会变长而使得界面更拥挤，甚至引起布局变化，此时就需要对超长的文字进行折行显示。

通常，CSS样式默认设定为按钮文字不允许折行。如果需要将按钮文字折行显示，可以重新定义ui-btn-inner样式为：

```
<style type="text/css">
  .ui-btn-inner {
    white-space: normal !important;
  }
</style>
```

在CSS中，!important表示CSS优先级，这里用以提升white-space:normal的优先级，从而使得按钮文字支持折行显示。

图7-10是应用折行显示前后的效果对比。

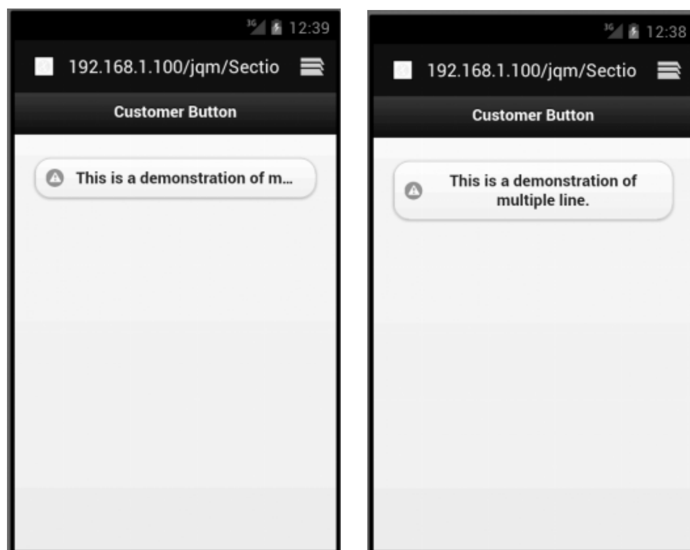


图7-10　左图为非折行显示，右图为折行显示的按钮样式

第 8 章 工具栏

在Web移动应用开发中，工具栏是一种常用的界面设计方式，一些常用的页面元素会集成在工具栏中，例如标题、后退按钮或者某些常用的链接。

在这一章中，我们将会了解到：

- 工具栏显示模式；
- 页眉和页脚工具栏；
- 导航工具栏；
- 固定工具栏及其属性、选项、方法和事件；
- 高级开发技巧。

8.1 工具栏显示模式

工具栏主要分为页脚工具栏和页眉工具栏。页脚工具栏更方便单手触控操作，所以在很多移动应用中，很多常用功能会放在这个工具栏中，而页眉工具栏会作为标题栏和导航功能使用。常见的jQuery Mobile工具栏以固定工具栏或者内联工具栏的形式呈现，如图8-1所示。



图8-1　左图为固定模式的工具栏，右图为内联模式的工具栏

在固定模式的工具栏中，当用户轻击移动设备浏览器时，会显示或者隐藏工具栏。固定工具栏在浏览器屏幕中的位置也是固定的，页眉工具栏总是位于浏览器屏幕最上方，而页脚工具栏总是处于浏览器屏幕最下方。

尽管固定模式的工具栏的应用更加普遍一些，但是默认情况下jQuery Mobile的工具栏不会被设定为固定模式的工具栏，而是以内联方式呈现在界面上。在内联模式的工具栏中，页眉工具栏将出现在页面正文内容的上方，而正文结束之后紧跟着的是页脚工具栏，并且随着正文内容的长短，工具栏的位置也会发生变化。

在jQuery Mobile中，很容易实现页眉工具栏和页脚工具栏。要实现页眉工具栏，开发者只需在div容器中添加data-role属性为header即可。同样，页脚工具栏所对应的属性data-role为footer。

如果工具栏被设置为固定模式，则每次轻击浏览器，工具栏就会被显示或者隐藏。如果工具栏是内联模式，则任何时候工具栏都会被呈现在页面的最上方和最下方。

默认情况下，页眉和页脚工具栏是以内联模式呈现的。如果需要以固定模式呈现工具栏，则需要添加data-position="fixed"属性。

图8-1左图所示的固定模式的工具栏的代码如代码清单8-1所示。在这段代码中，页眉工具栏所在的div容器的data-role属性被设置为header，页脚工具栏的div容器的data-role属性被设置为footer，并且页眉工具栏和页脚工具栏都设置属性data-position为fixed。

代码清单8-1　固定模式的工具栏的代码示例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="工具栏">
      <div data-role="header" data-position="fixed" data-fullscreen="true" data-theme="c">
        <h2>页眉工具栏</h2>
      </div>
      <h1>工具栏</h1>
      页眉和页脚工具栏有助于使用者进行常用操作。
      <div data-role="footer" data-position="fixed" data-theme="c">
        <h2>页脚工具栏</h2>
      </div>
    </section>
  </body>
</html>
```

注意 即便页眉或者页脚工具栏中只包含文字，我们也建议开发者加上标题标签，并且任何标题标签（h1到h6）都可以。如果不加入标题标签，则工具栏所呈现的高度、字号和边距等可能出现显示异常。

8.2 页眉和页脚工具栏

对于页眉工具栏，我们主要设置工具栏中的按钮样式和位置。

注意 页眉工具栏按钮通常是超级链接按钮，而不是表单按钮，用错的话可能导致页面呈现异常与功能异常。

在页眉工具栏中，按钮默认情况下分立在标题文字的两端。左侧按钮会在标题栏中居左显示，右侧按钮则会居右显示。默认情况下，页眉工具栏中的按钮都将呈现为内联样式。

代码清单8-2演示了在页眉工具栏中顺序放置按钮和标题文字的代码。

代码清单8-2 在页眉工具栏中顺序放置按钮和标题文字

```
<div data-role="header" data-position="fixed" data-fullscreen="true">
  <a href="#" data-role="button" data-icon="arrow-l" data-theme="c">
    左侧
  </a>
  <h2>页眉工具栏</h2>
  <a href="#" data-role="button" data-icon="arrow-r" data-theme="c">
    右侧
  </a>
</div>
```

运行上述代码，得到的运行结果如图8-2所示。



图8-2 页眉按钮位置

如果需要指定页眉按钮位于页眉的一侧，可以通过定义CSS样式来实现。表8-1列出了设置工具栏中按钮位置的样式属性。

表8-1　样式属性

样式名称	作　用
.ui-btn-left	按钮位于页眉左侧
.ui-btn-right	按钮位于页眉右侧

如果页眉工具栏中只有图标而没有标题文字，也不能单纯地只在其中放入超级链接按钮。如果没有标题标签h1到h6，页眉高度将会显示异常，此时可以通过添加一个DOM容器，并设置class属性为ui-title来解决：

```
<span class="ui-title"></span>
```

这个DOM容器的作用就是撑开页眉工具栏，使之不会变形。尽管有这样的处理页面变形方法，我们依然建议开发者：如果工具栏中包含有标题文字，还是要将标题文字放入h1到h6的标题中。

如果希望在页眉工具栏中添加返回按钮，可以在超级链接按钮中添加data-add-back-btn="true"属性，此时按钮中原有的超级链接指向将不再起作用，而只保留返回功能。作为一种经验，尽管添加了data-add-back-btn属性，按钮中依然建议保留指向前一个页面的超级链接地址。这是因为一些浏览器可能不支持data-add-back-btn属性，此时依然可以通过定义的超级链接地址而回到正确的页面。

页脚工具栏和页眉工具栏的按钮设置大致相同。从用户行为习惯出发，按钮组和表单也可以放置在页脚工具栏，以便用户操作。

注意　如果在页脚工具栏中使用按钮组，需要将按钮组设置为水平排列按钮的方式。默认情况下，按钮组将垂直排列各个按钮。

设置按钮组为水平排列，可以通过在data-role="controlgroup"的按钮组DOM容器中添加属性data-type="horizontal"来实现。

8.3　导航工具栏

导航工具栏可以容纳一些导航按钮，用以控制页面或功能间切换。导航工具栏通常与页眉工具栏或页脚工具栏共同使用，可以理解为页眉或页脚工具栏在导航功能上的补充。有别于通常所见的按钮样式，导航工具栏按钮没有了圆形的按钮边界，只有方正的导航按钮。如果导航工具栏中有两个导航按钮，则每个按钮将各占移动设备浏览器宽度的一半。

代码清单8-3展示了实现导航工具栏的过程，大致分为下面这两个步骤。

(1) 在工具栏容器中构造包含data-role="navbar"属性的div容器，而这个div容器将作为导航工具栏的容器使用。

(2) 在这个导航工具栏的div容器中，加入经过
列表标签包装的超级链接。

代码清单8-3　导航工具栏示例

```
<div data-role="navbar">
  <ul>
    <li><a href="#">导航按钮</a></li>
    <li><a href="#">导航按钮</a></li>
    <li><a href="#">导航按钮</a></li>
    <li><a href="#">导航按钮</a></li>
  </ul>
</div>
```

注意　虽然导航工具栏看似一组横向排列的按钮，但是这些按钮并不是Form按钮，而是按钮样式的超级链接。

如果希望导航按钮中包含按钮图标，可以通过添加data-icon属性来实现，如代码清单8-4所示。

代码清单8-4　包含图标按钮的导航工具栏

```
<div data-role="navbar">
  <ul>
    <li><a href="#" data-icon="arrow-l">向前</a></li>
    <li><a href="#" data-icon="arrow-r">向后</a></li>
  </ul>
</div>
```

若要集成导航工具栏与页眉或页脚工具栏，只需要将导航工具栏嵌入到页眉或页脚工具栏即可，示例代码如代码清单8-5所示。导航工具栏可以单独通过data-theme来美化样式，相关内容可参见第13章。

代码清单8-5　集成在页眉和页脚工具栏的导航工具栏

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="导航工具条">
```

```
      <header data-role="header" data-position="fixed">
        <h1>导航工具栏</h1>
        <div data-role="navbar">
          <ul>
            <li><a href="#">导航按钮</a></li>
            <li><a href="#">导航按钮</a></li>
            <li><a href="#">导航按钮</a></li>
            <li><a href="#">导航按钮</a></li>
          </ul>
        </div>
      </header>
      <div data-role="content">
        <p>页眉或页脚导航工具栏都可以集成导航工具栏以实现丰富的导航效果。
        </p>
      </div>
      <div data-role="footer" data-position="fixed">
        <div data-role="navbar">
          <ul>
            <li><a href="#" data-icon="arrow-l">向前</a></li>
            <li><a href="#" data-icon="arrow-r" >向后</a></li>
          </ul>
        </div>
      </div>
    </section>
  </body>
</html>
```

运行上述代码，得到的运行效果如图8-3所示。



图8-3　导航工具栏

在传统网页呈现中，如果内容超出一屏显示范围，那么将会在浏览器右侧出现一个垂直滚动条，而在移动浏览器中却不会。当用手向上滑动屏幕的时候，内容将会随之移动。由于页眉和页脚工具栏被添加了data-position="fixed"属性，所以内容在上下翻动的时候，导航工具栏的位置是不变的，这样用户就可以方便地点击导航按钮进行相应操作。

在导航工具栏中，每行最多有5个导航按钮，超过则自动换行显示。在自动换行的导航工具栏中，每行只有两个导航按钮，如图8-4所示。



图8-4　包含7个导航按钮的导航工具栏

8.4　固定工具栏属性、选项、方法和事件

在前面的介绍中，我们初步了解到固定工具栏的特点。我们可以将页眉或页脚工具栏设置为固定工具栏，当页面中的内容被拖动时，固定导航栏的位置依然保持停靠屏幕上下，而不会发生变动。

固定工具栏可以通过设置属性和选项来设定呈现形式，也可以通过方法和事件进行交互操作，下面我们就来简单介绍一下。

8.4.1　属性

使用固定工具栏的属性，可以设定固定工具栏的呈现或操作效果。常用的固定工具栏属性如表8-2所示。

表8-2　固定工具栏属性

属　　性	含　　义
data-visible-on-page-show	设置页面被加载时，是否显示固定工具栏。 ● 默认为true，自动呈现固定工具栏。 ● 设置为false时，则隐藏固定工具栏。 如果设置data-visible-on-page-show属性，通常会一块设置页眉工具栏和页脚工具栏，否则每次轻击屏幕，一个工具栏被隐藏而另一个则被打开。 示例代码： <div data-role="footer" data-position="fixed" data-visible-on-page-show="false">…

（续）

属　性	含　义
data-disable-page-zoom	设置页面是否允许缩放。 ● 默认为true，不允许对页面进行缩放。 ●设置为false时，则允许对页面进行缩放。 示例代码： <div data-role="footer" data-position="fixed" data-disable-page-zoom="false">…
data-transition	设置工具栏切换方式。 ● 默认为幻灯方式slide。 ●设置为fade时，为淡入淡出效果。 ● 设置为none时，为无动画效果。 在一些操作系统上，data-transition并不如预期的这样显示。 示例代码： <div data-role="footer" data-position="fixed" data-transition="slide">…
data-fullscreen	设置以全屏方式显示固定工具栏。 示例代码： <div data-role="footer" data-position="fixed" data-fullscreen ="false">…
data-tap-toggle	设置屏幕轻击之后是否隐藏与显示。 ● 默认为true，轻击或用鼠标点击屏幕时，显示或隐藏固定工具栏。 ● 设置为false时，当轻击或者用鼠标点击屏幕时，固定工具栏始终不变。如果之前显示，则始终显示，反之亦然。 示例代码： <div data-role="footer" data-position="fixed" data-tap-toggle="false">…
data-update-page-padding	设置固定工具栏的页面填充，默认值为true。如果设置为false，则可能在方向切换或其他尺寸调整中不会更新页面填充尺寸

8.4.2　选项

选项用以通过程序对固定工具栏进行设定。由于大多数固定工具栏选项和属性的使用非常类似，表8-3列出了选项与属性的对照表。

表8-3　固定工具栏选项与属性对照表

选　项	属　性	功　能
visibleOnPageShow	data-visible-on-page-show	设置页面被加载时，是否显示固定工具栏
disablePageZoom	data-disable-page-zoom	设置页面是否允许缩放
transition	data-transition	设置工具栏切换方式
fullscreen	data-fullscreen	设置以全屏方式显示固定工具栏
tapToggle	data-tap-toggle	设置屏幕轻击之后是否隐藏与显示
updatePagePadding	data-update-page-padding	设置固定工具栏的页面填充

除去上述与固定工具栏属性功能接近的选项之外，固定工具栏还有一些选项没有对应的属

性，使用过程中需要通过JavaScript对其进行控制，这些选项如表8-4所示。

表8-4 固定工具栏选项

属 性	含 义
tapToggleBlacklist	如果轻击在特定CSS样式之上，则不会隐藏或展开固定工具栏。 默认值为"a, .ui-header-fixed, .ui-footer-fixed" 下面这段代码用于实现轻击在input对象上，不会隐藏或展开固定工具栏： $(".headerToolbar").fixedtoolbar({ tapToggleBlacklist: "a, input, .ui-header-fixed, .ui-footer-fixed" });
hideDuringFocus	如果焦点落在特定DOM对象上，则自动隐藏固定工具栏。 默认值为"input, select, textarea" 下面这段代码实现焦点落在input上时自动隐藏固定工具栏： $(".headerToolbar").fixedtoolbar({ hideDuringFocus: "input" });
supportBlacklist	反馈是否支持黑名单的布尔数值。
initSelector	自定义CSS样式名称，用以声明固定工具栏。 默认值为：":jqmData(position='fixed')"，表示在包含有data-role属性值为header或者footer的容器中声明属性data-position为fixed，则这个容器为固定工具栏

8.4.3 方法和事件

JavaScript可以通过方法对固定工具栏进行展开、隐藏和销毁等操作，具体如表8-5所示。

表8-5 固定工具栏方法及其含义

方 法	含 义
show	打开指定的固定工具栏。 示例代码： $("#footerToolbar").fixedtoolbar('show');
hide	隐藏指定的固定工具栏。 示例代码： $("#footerToolbar").fixedtoolbar('hide');
toggle	切换固定工具栏的显示或隐藏状态。 示例代码： $("#footerToolbar").fixedtoolbar('toggle');
updatePagePadding	更新页面填充。 示例代码： $("#footerToolbar").fixedtoolbar('updatePagePadding');
destroy	恢复固定工具栏元素到初始状态。注意，这里不是销毁或者删除。 示例代码： $("#footerToolbar").fixedtoolbar('destory');

注意 只有页面的内容高度超过屏幕高度时，固定工具栏的这些方法的使用效果才会表现出来，否则固定工具栏将始终出现在浏览器上，而不会消失。

代码清单8-6展示了固定工具栏方法的使用方法，包括显示、隐藏、状态切换、更新填充、恢复到初始状态等操作。

代码清单8-6　固定工具栏方法的使用示例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type="text/javascript">
      $(document).ready(function(e) {
        $("#footerToolbar").fixedtoolbar({tapToggleBlacklist: "a, button, input,
          select, textarea, .ui-header-fixed, .ui-footer-fixed" });
      });
      function btnShowToolbar(){
        $("#footerToolbar").fixedtoolbar('show');
      }
      function btnHideToolbar(){
        $("#footerToolbar").fixedtoolbar('hide');
      }
      function btnToggleToolbar(){
        $("#footerToolbar").fixedtoolbar('toggle');
      }
      function btnUpdatePagePaddingToolbar(){
        $("#footerToolbar").fixedtoolbar('updatePagePadding');
      }
      function btnDestoryToolbar(){
        $("#footerToolbar").fixedtoolbar('destory');
      }
    </script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="导航工具栏">
      <div data-role="header" data-position="fixed">
        <h1>固定工具栏方法</h1>
      </div>
      <div data-role="content">
        <p>通过自定义方法对固定工具栏执行操作。
          <button onClick="btnShowToolbar();">Show方法</button>
          <button onClick="btnHideToolbar();">Hide方法</button>
          <button onClick="btnToggleToolbar();">Toggle方法</button>
          <button onClick="btnUpdatePagePaddingToolbar();">UpdatePagePadding方法</button>
          <button onClick="btnDestoryToolbar();">Destory方法</button>
        </p>
      </div>
      <div data-role="footer" id="footerToolbar" data-position="fixed">
        <h2>页脚工具栏</h2>
      </div>
```

```
    </section>
  </body>
</html>
```

需要注意的是，在这个代码示例中，各个固定工具栏方法是通过按钮来触发的，而按钮并不在固定工具栏tapToggleBlacklist选项默认设定的范围之中，这也就意味着每次触碰按钮时，除了执行固定工具栏方法外，还会触发固定工具栏显示或者隐藏。为此，在程序中特别对tapToggleBlacklist进行扩展，将按钮、输入框、选择框和文本框等元素也添加到黑名单中，以保证应用正确执行。

此外，需要说明的是，当建立固定工具栏的时候，将会触发create事件。

8.5　高级开发技巧

工具栏的开发，通常主要是固定工具栏的开发。jQuery Mobile库为开发者提供了默认的图标与预定义的风格样式，开发者可以根据业务场景的需要对固定工具栏的样式、风格、图标和文字进行定制。

8.5.1　自定义图标导航工具栏

导航工具栏为开发者提供了一套用户界面风格统一、导航按钮尺寸均匀的工具栏。只是jQuery Mobile所能提供的按钮种类相对有限，内置的图标数量也相对有限。通过自定义按钮图标，扩展既有界面呈现样式将有助于开发语义更加清晰、便于使用的用户界面。

开发自定义图标导航工具栏，大致包括这样几个主要的定制化步骤。

(1) 设定导航按钮与图标背景样式。

(2) 选取自定义导航按钮。

(3) 将定制导航按钮集成到导航工具栏。

代码清单8-7是自定义图标导航工具栏的示例。

代码清单8-7　自定义工具栏图标代码示例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <style type="text/css">
    .nav-glyphish .ui-btn .ui-btn-inner {
      padding-top: 43px !important;
    }
    .nav-glyphish .ui-btn .ui-icon {
```

```
        width: 45px!important;
        height: 35px!important;
        margin-left: -24px !important;
        box-shadow: none!important;
        -moz-box-shadow: none!important;
        -webkit-box-shadow: none!important;
        -webkit-border-radius: none !important;
        border-radius: none !important;
      }
      #favorite .ui-icon {
        background-image: url(http://glyphish.com/images/demo.png);
        background-position: -345px -112px;
        background-repeat: no-repeat;
      }
      #recent .ui-icon {
        background-image: url(http://glyphish.com/images/demo.png);
        background-position: -9px -61px;
        background-repeat: no-repeat;
      }
      #contacts .ui-icon {
        background-image: url(http://glyphish.com/images/demo.png);
        background-position: -9px -540px;
        background-repeat: no-repeat;
      }
      #voicemail .ui-icon {
        background-image: url(http://glyphish.com/images/demo.png);
        background-position: -394px -733px;
        background-repeat: no-repeat;
      }
      </style>
    </head>
    <body>
      <section id="MainPage" data-role="page" data-title="导航工具栏">
        <header data-role="header" data-position="fixed">
          <h1>导航工具栏</h1>
          </header>
        <div data-role="content">
          <p>自定义图标的导航工具栏。
          </p>
        </div>
        <div data-role="footer" class="nav-glyphish" data-position="fixed">
          <div data-role="navbar" class="nav-glyphish" data-grid="c">
            <ul>
              <li><a href="#" id="favorite" data-icon="custom">收藏</a></li>
              <li><a href="#" id="recent" data-icon="custom">最近</a></li>
              <li><a href="#" id="contacts" data-icon="custom">联系人</a></li>
              <li><a href="#" id="voicemail" data-icon="custom">录音</a></li>
            </ul>
          </div>
        </div>
      </section>
    </body>
  </html>
```

运行上述代码，得到的自定义图标呈现效果如图8-5所示。



图8-5 自定义图标的导航工具栏

代码中的CSS实现了对于自定义图标的样式定义。

首先，对按钮的高度、尺寸和边框等进行定义，这样的定义包括padding尺寸、图标高度、宽度和边界等，并通过声明!important优先级确保设置的样式有效，具体代码如下：

```
.nav-glyphish .ui-btn .ui-btn-inner {
  padding-top: 43px !important;
}

.nav-glyphish .ui-btn .ui-icon {
  width: 45px!important;
  height: 35px!important;
  margin-left: -24px !important;
  box-shadow: none!important;
  -moz-box-shadow: none!important;
  -webkit-box-shadow: none!important;
  -webkit-border-radius: none !important;
  border-radius: none !important;
}
```

然后，针对不同按钮功能逐一设置不同的图标，这样的定义包括背景图片CSS样式名称、背景图片名称和位置等，具体代码如下：

```
#favorite .ui-icon {
  background-image: url(http://glyphish.com/images/demo.png);
  background-position: -345px -112px;
  background-repeat: no-repeat;
}
```

最后，在导航工具栏的超级链接按钮上设置定义了背景图片和显示尺寸的CSS样式：

```
<li><a href="#" id="favorite" data-icon="custom">收藏</a></li>
```

这里的图标取自http://glyphish.com。如果工程师需要使用这个网站的图标，最好下载到国内的服务器，以保证图片加载速度。

8.5.2　定制风格导航工具栏

在前一节中，我们实现了自定义图标的设定与个性化定制。在本节中，我们不但支持自定义风格导航工具栏，而且可以通过浮于其上的提示消息提供更多信息给使用者，示例图如图8-6所示。



图8-6　自定义风格导航工具栏

这里所使用的定制风格导航工具栏并不是jQuery Mobile自带的导航工具栏的样式，而是通过开源项目Bartender（其下载地址为https://github.com/frequent/bartender/）对jQuery Mobile进行扩展而实现的。Bartender的核心为一个CSS样式表，对jQuery Mobile的样式定义进行扩展，使得导航工具栏具有更强的用户界面效果。

项目的演示地址位于http://www.stokkers.mobi/valuables/bartender.html。

要在项目中使用Bartender，大致需要如下3个步骤。

(1) 定制Bartender项目中bartender.css所引用的图标与`data-icon`定义。

(2) 将bartender.css引用到需要支持这种效果的页面。

(3) 使用Bartender项目风格定义导航工具栏。

在Bartender项目中，分别定义了高分辨率图片和低分辨率图片的样式，如代码清单8-8所示。

代码清单8-8　高分辨率与低分辨率背景图片设置

```
.hi-res {
  width: 30px;
  height: 30px;
  background: url("examples/sprite_hi-res.png") no-repeat;
```

```
  margin-left: 70px;
  background-size: 150px 44px;
  -o-background-size: 150px 44px;
  -webkit-background-size: 150px 44px;
  -moz-background-size: 150px 44px;
  -ms-background-size: 150px 44px;
}

.lo-res {
  width: 30px;
  height: 30px;
  background: url("examples/sprite_lo-res.png") no-repeat;
  margin: -30px 0 0 20px;
}
```

在低分辨率环境下，图片以代码清单8-9所示的方式被分隔开，并定义在不同的data-icon中。

代码清单8-9　在低分辨率下，不同CSS样式定义不同的分隔图片和位置

```
.soloSprite li a .ui-btn-inner {
  display: inline-block;
  position: static;
  height: 30px;
  width: 30px;
  background-color: none;
  background: url("sprite_lo-res.png") no-repeat;
  background-size: 300px 44px;
  -o-background-size: 300px 44px;
  -webkit-background-size: 300px 44px;
  -moz-background-size: 300px 44px;
  -ms-background-size: 300px 44px;
}

.soloSprite li a[data-icon="features"] span:only-child {
  background-position: 0px 0px
}

/* 此处隐去其他data-icon定义*/
...
```

在高分辨率环境下，CSS将加载和处理高分辨率图片，如代码清单8-10所示。

代码清单8-10　高分辨率下，不同CSS样式定义不同的分隔图片和位置

```
@media only screen and (-webkit-min-device-pixel-ratio: 1.5),  only screen and
  (min--moz-device-pixel-ratio: 1.5),  only screen and (min-resolution: 240dpi) {
  .soloSprite li a .ui-btn-inner {
    display: inline-block;
    position: static;
    height: 30px;
    width: 30px;
    background-color: none;
    background: url("sprite_hi-res.png") no-repeat;
    background-size: 150px 66px;
```

```
    -o-background-size: 150px 66px;
    -webkit-background-size: 150px 66px;
    -moz-background-size: 150px 66px;
    -ms-background-size: 150px 66px;
  }
  .soloSprite li a[data-icon="features"] span:only-child {
    background-position:0px -22px
  }

/* 此处隐去其他data-icon定义*/
...

}
```

小技巧 随着移动设备尺寸和显示分辨率的多样化，在基于HTML5和jQuery Mobile所开发的应用中，将会越来越多地使用媒体查询（media query）技术，建议开发者在开发过程中能够熟练掌握这个技术。

有别于常见的jQuery插件，如果希望基于这套定制风格的导航工具栏插件加载更多的图标，那么需要修改bartender.css。将预先定义好的图标文件设置在相应的CSS位置，特别是如果需要支持不同分辨率的移动设备浏览器，那么最好也能设计一套高分辨率的图标，以获得更好的用户体验。

在开发这样的导航工具栏中，和之前所见的其他导航工具栏一样，将导航工具栏放在一个页脚工具栏中。在ul的CSS定义中添加apple-navbar-ui和comboSprite属性，此时具有自定义按钮和适合不同屏幕分辨率的导航工具栏就初具形态了。如果需要在导航工具栏中添加气泡提示，例如新消息数量或者一段简短提醒文字，则可以在new中进行定义。

代码清单8-11是经由Bartender项目美化的导航工具栏代码片段。

代码清单8-11 集成自定义图标到导航栏，并设置气泡提示new字样

```
<li>
  <a href="#features" data-iconpos="top" data-icon="features">
    特性<span class="ui-li-count">new</span>
  </a>
</li>
```

使用者在基于Bartender项目开发自己的导航工具栏时，需要注意下面这些细节。

- 在Bartender项目中，图片背景文件sprite_hi-res.png和sprite_lo-res.png只包含用于演示的少量图标。在开发者使用的过程中，需要结合自己的需要制作合适的图标。
- Bartender项目通过媒体查询在不同移动设备分辨率下加载不同的样式和图片。作为一种有益的经验，开发者可以保留或增强这样的实现。
- data-icon属性的CSS定义位于Bartender.CSS中，使用者需要小心定制此处的定义。

完整的制定气泡提示的导航工具栏代码如代码清单8-12所示。

代码清单8-12　定制风格导航工具栏的完整的HTML代码示例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <link rel="stylesheet" href="bartender/bartender.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="自定义风格导航工具栏">
      <header data-role="header" data-position="fixed">
        <h1>自定义风格工具栏</h1>
      </header>
      <div data-role="content">
      </div>
      <div data-role="footer" data-position="fixed">
        <div data-role="navbar" data-grid="d">
          <ul class="apple-navbar-uicomboSprite">
            <li>
              <a href="#features" data-iconpos="top" data-icon="features">
                特性<span class="ui-li-count">new</span>
              </a>
            </li>
            <li>
              <a href="#brands" data-iconpos="top" data-icon="brands">
                品类<span class="ui-li-count">10</span>
              </a>
            </li>
            <li>
              <a href="#fees" data-iconpos="top" data-icon="fees">
                费率
              </a>
            </li>
            <li>
              <a href="#contact" data-iconpos="top" data-icon="contact">
                联系
              </a>
            </li>
            <li>
              <a href="#about" data-iconpos="top" data-icon="about">
                关于我们
              </a>
            </li>
          </ul>
        </div>
      </div>
```

```
    </section>
  </body>
</html>
```

注意　单纯的HTML文件是不能独立实现定制风格导航工具栏的。如果需要实现完整功能，则需要手工修改bartender.css文件。

定制风格导航工具栏时，按钮的背景图片需要在CSS中定义。这就需要修改bartender.css文件，这个修改是重点所在。具体的修改方法可参见代码清单8-8、代码清单8-9和代码清单8-10的相关介绍。

第 9 章 列表视图

9

列表视图经常用于移动应用的列表内容管理，它通过列表的方式将内容有序地排列和管理起来。此外，列表视图也可以作为页面导航使用。在有限的移动设备屏幕尺寸下，使用者可以使用列表视图方便地实现导航菜单的功能。

在这一章中，我们将会了解到：

- 基本概念；
- 嵌套列表；
- 分类列表；
- 数字列表；
- 分立按钮列表；
- 缩略图与图标列表；
- 气泡提示；
- 只读列表；
- 过滤列表内容；
- 插页列表；
- 折叠列表；
- 自动分类列表视图；
- 使用列表美化表单布局；
- 列表内容排版美化；
- 列表视图属性、选项、方法和事件；
- 高级编程技巧 。

9.1 基本概念

如果以单纯的HTML实现一段列表而言，使用jQuery Mobile实现列表的方法和面对桌面应用的HTML页面所实现列表的方法相差无几。对于使用jQuery Mobile实现移动应用列表而言，简单地说，只需要在列表DOM容器上设置`data-role`属性为`listview`，列表将会被呈现为移动应用风格的样式。

图9-1对比了jQuery Mobile风格的列表视图与传统列表视图。之所以会有这样的不同，是因为在jQuery Mobile风格的页面中，列表DOM容器的`data-role`属性被设置为`listview`。图9-1左图的代码如代码清单9-1所示。



图9-1　左图为jQuery Mobile风格列表视图，右图为传统列表视图

代码清单9-1　jQuery Mobile列表视图

```
<ul data-role="listview">
  <li>北京</li>
  <li>天津</li>
  <li>上海</li>
  <li>重庆</li>
</ul>
```

在很多实际应用场景中，列表视图也可以作为导航来使用。如果需要列表视图具有导航功能，直接在列表项中加入相应超级链接即可，代码清单9-2展示了相关示例。

代码清单9-2　导航列表视图

```
<ul data-role="listview">
  <li><a href="http://jquerymobile.com" rel="external">jQuery Mobile</a></li>
  <li><a href="http://www.ituring.com.cn" rel="external">图灵社区</a></li>
  <li><a href="http://catchexception.net" rel="external">CatchException.NET</a></li>
</ul>
```

运行上述代码，得到的运行结果如图9-2所示。

在包含有超级链接的列表视图中，在超级链接的右侧默认会出现一个向右的箭头图标，以表示这个列表条目是一个超级链接。



图9-2 导航列表视图

9.2 嵌套列表

顾名思义，嵌套列表指的是多于一个层次的列表，一级列表之下包含着二级或更多级的列表。这样的嵌套可以是一层，也可以是多层。通常，嵌套深度不会很大，否则会影响逐层进入和逐层返回的用户体验。

如果想开发嵌套列表，只需在之前所介绍的列表视图的基础上嵌套新的一层列表即可，而下一级的嵌套列表将会继承上一级的属性设置。例如，上一级的列表视图设置了某个主题样式，那么下一级所嵌套的列表将会继承这些主题样式的设置。

代码清单9-3在之前列表视图的基础之上实现了嵌套列表的功能。

代码清单9-3 嵌套列表

```
<ul data-role="listview">
  <li>北京市
    <ul>
      <li>朝阳区</li>
      <li>东城区</li>
      <li>西城区</li>
    </ul>
  </li>
  <li>天津市
    <ul>
      <li>河东区</li>
      <li>和平区</li>
      <li>南开区</li>
    </ul>
  </li>
</ul>
```

运行上述代码，得到的结果如图9-3左图所示，而图9-3右图呈现了传统Web桌面应用的嵌套列表样式。

在这个jQuery Mobile嵌套列表视图中，每个城市嵌套有下一级的区县列表。可以发现，每个列表条目右侧出现了指向下一级列表的向右箭头图标。



图9-3　左图为移动应用列表视图，右图为桌面Web应用的列表

点击“北京市”列表项，将会进入下一级区县列表，如图9-4所示。



图9-4　点击“北京市”进入的区县列表页

在嵌套列表中，如果要从下级列表返回到上一级，在Android系统中可以直接使用手机下方的返回键。但是对于iOS操作系统的浏览器来说，就没那么方便了，此时开发者可以通过加入触控操作实现返回或者跳转到后续页面，如轻扫或者按钮，相关内容可参见9.16节。

9.3　分类列表

分类列表通过分类标记将不同类别的内容集中放在一个列表中，示例图如图9-5所示。



图9-5 分类列表

分类列表视图是基于基本的列表视图增加分类标签而实现的，分类标签也是列表的一部分。将包含有分类信息的文字放置于`<li data-role="list-divider">`标签中，就实现了分类标签的定义。在分类列表中，其他部分的内容和之前所介绍的列表视图是一样的。

前面所展示的分类列表视图的代码如代码清单9-4所示。在这个代码片段中，大洋和大陆分别是两个分类标签，我们将这两个分类标签的文字分别放在`data-role`属性为`list-divider`的列表项中。

代码清单9-4 分类列表的代码片段

```
<ul data-role="listview">
  <li data-role="list-divider">大洋</li>
  <li>太平洋</li>
  <li>大西洋</li>
  <li>印度洋</li>
  <li>北冰洋</li>
  <li data-role="list-divider">大陆</li>
  <li>亚欧大陆</li>
  <li>非洲大陆</li>
  <li>南美大陆</li>
  <li>北美大陆</li>
  <li>南极大陆</li>
  <li>澳大利亚大陆</li>
</ul>
```

9.4 数字列表

数字列表的主要特点在于，在每个列表项之前呈现序数标记，如图9-6所示。数字从1开始，

自上而下依次递增。声明数字列表时，需要将 HTML 标记定义为`ol`（Ordered List，有序列表），这有别与之前章节中使用`ul`（Unordered List，无序列表）标签实现列表功能。



图9-6　左图为jQuery Mobile风格的数字列表视图，右图为传统数字列表

为使jQuery Mobile样式应用于数字列表，同其他列表一样，需要在`ol`标签中声明`data-role`属性为`listview`。

代码清单9-5是数字列表的代码片段。

代码清单9-5　数字列表

```
<ol data-role="listview">
  <li>北京市</li>
  <li>天津市</li>
  <li>上海市</li>
  <li>重庆市</li>
</ol>
```

9.5　分立按钮列表

分立按钮列表是jQuery Mobile列表的一种排版样式，效果如图9-7所示。在分立按钮列表中，每个列表分为两部分：前面一部分是惯常的列表内容，后面一部分位于列表页面的右侧，它是独立的一列，包含有图标按钮。

列表前半部分的文字和后半部分的图标可以是相同的超级链接，也可以是不同的。这没有一定之规，完全是基于应用的使用场景而定的。

分立按钮列表的代码如代码清单9-6所示。



图9-7 分立按钮列表

代码清单9-6 分立按钮列表

```
<ul data-role="listview">
  <li><a href="#">北京市</a><a href="#" data-icon="plus">更多</a></li>
  <li><a href="#">天津市</a><a href="#" data-icon="plus">更多</a></li>
  <li><a href="#">上海市</a><a href="#" data-icon="plus">更多</a></li>
  <li><a href="#">重庆市</a><a href="#" data-icon="plus">更多</a></li>
</ul>
```

在开发分立按钮列表的时候，开发者需要注意两个细节。

- 在上面的示例代码中，左侧文字部分一定要使用超级链接标签包含文字内容，否则可能引起排版错位。
- 不建议在列表中放置过长的文字。列表文字过长，则可能会被截断，此时虽然可以加入换行以保证文字显示完整，但是，不同移动设备的屏幕尺寸不同，每行文字的字数也不尽相同，强制换行也可能导致界面排版混乱。

9

小技巧 jQuery Mobile为按钮列表提供了一些默认的图标，开发者既可以使用这些图标，也可以根据Web移动应用而使用自定义的图标按钮。如果希望使用自定义的按钮图标，请参见第7章。

9.6 缩略图与图标列表

缩略图列表是指在列表项文字的前方包含有一个缩略图，实现时，只需在列表项文字之前加入缩略图即可。

实现缩略图的代码如代码清单9-7所示。

代码清单9-7　缩略图列表视图

```
<ul data-role="listview">
  <li>
    <a href="#"><img src="image/Beijing.jpg"/>北京市</a>
  </li>
  <li>
    <a href="#"><img src="image/Tianjin.jpg"/>天津市</a>
  </li>
  <li>
    <a href="#"><img src="image/Shanghai.jpg"/>上海市</a>
  </li>
  <li>
    <a href="#"><img src="image/Chongqing.jpg"/>重庆市</a>
  </li>
</ul>
```

在列表中使用图标与使用缩略图的方法大致相当，只需要在图标文件中添加class="ui-li-icon"即可，例如，

```
<a href="#"><img src="icon/bj-icon.jpg" class="ui-li-icon"/>北京市</a>
```

从用户体验设计的角度而言，在列表视图中使用图标和缩略图存在一些细微的差别，具体如下所示。

- 图标列表中的图标向右和向下缩进更多。
- 在图标列表中，图标尺寸通常更小，不会撑高列表。在缩略图列表中，如果缩略图高度较大，则会撑高列表。

9.7　气泡提示

在进行列表视图呈现的时候，可以加入提示数据或者一段短小的提示消息，用以指导用户操作。气泡提示的呈现效果如图9-8所示。

实现气泡提示时，需要在列表的基础上完成两个步骤。

(1) 将列表内容文字置于超级链接标签<a>之中，例如 列表项文字。

(2) 在超级链接标签<a>内部、列表文字之后添加气泡提示标签 气泡提示内容：

```
<li><a href="#">特价展卖<span class="ui-li-count">new</span></a>
```

代码清单9-8是实现气泡提示的代码片段。



图9-8　气泡提示列表

代码清单9-8　气泡提示列表视图

```
<ul data-role="listview">
  <li>
    <a href="#">新品上架<span class="ui-li-count">new</span></a>
    <ul>
      <li>电器</li>
      <li>数码</li>
      <li>图书</li>
      <li>家居</li>
    </ul>
  </li>
  <li>
    <a href="#">特价展卖<span class="ui-li-count">90</span></a>
  </li>
</ul>
```

气泡提示的内容既可以是数字，也可以是文字。如果是一段文字提示，通常只是短语，不建议放置大段文字。如果文字过长，因为移动设备的屏幕尺寸较小，界面会很难看。

不建议在嵌套列表中使用气泡提示。如果使用嵌套列表，将会在下一级嵌套列表中显示气泡提示文字。这个提示文字的显示将会使得嵌套列表的标题看上去不知所云。如图9-9所示，在嵌套列表的标题中，新品上架是上一级列表的文字，new则是气泡提示的文字。在嵌套列表中，这两部分文字都被显示了出来。

如果需要同时实现嵌套列表与气泡提示的呈现效果，可以开发多个列表页面，通过超级链接而非嵌套列表的方式将这些列表组织起来。当点击一个列表中的超级链接时，页面将会跳转到新的页面。



图9-9　气泡提示与嵌套列表混合使用的效果

9.8　只读列表

只读列表的内容不包含任何超级链接，而只是单纯的列表功能。实现只读列表时，没有特别的属性声明，只是在列表中不再包含超级链接即可。

代码清单9-9是只读列表的代码示例。

代码清单9-9　只读列表视图

```
<ul data-role="listview">
  <li>北京</li>
  <li>天津</li>
  <li>上海</li>
  <li>重庆</li>
</ul>
```

只读列表所呈现的效果如图9-10所示。



图9-10　只读列表

9.9　过滤列表内容

受到移动设备界面尺寸的限制，当列表条目很多的时候，用户很难快速定位到列表内容，过滤列表内容就是为这种场景设计的。随着用户的输入，包含用户输入文字的条目会被自动检索和显示出来，不论列表条目有多少。这个时候，用户输入列表中可能包含的是一个字母、词根或者单词，列表自动过滤出包含这段文字的内容，并将显示范围缩小到一个更精准的范围。这样，使用者就可以在这个有限的范围内快速定位所感兴趣的内容。

在图9-11所示的这个演示界面中，使用者输入英文字母p时，列表视图自动将包含有字母p的流行的计算机语言检索出来。



图9-11　左图为流行的计算机语言列表，右图为包含字母p的语言

要实现过滤功能，需要在列表定义的DOM容器中声明`data-filter`属性为`true`。声明之后，jQuery Mobile将自动在列表开始的位置添加一个输入框，使用者可以基于这个输入框过滤列表中的内容，例如：

```
<ul data-role="listview" data-filter="true"> … </ul>
```

实现过滤列表功能的代码片段如代码清单9-10所示。

代码清单9-10　过滤列表视图

```
<ul data-role="listview" data-filter="true">
  <li>C</li>
  <li>Java</li>
  <li>C++</li>
  <li>Object-C</li>
  <li>C#</li>
```

```
    <li>PHP</li>
    <li>Basic</li>
    <li>Python</li>
    <li>JavaScript</li>
    <li>Perl</li>
    <li>Ruby</li>
    <li>PL/SQL</li>
    <li>Delphi</li>
    <li>Visual Basic.NET</li>
    <li>Lisp</li>
    <li>Pascal</li>
</ul>
```

如果列表中的内容比较多，即便使用过滤条件，依然需要使用者从大量列表条目中进行人工筛选。如果开发者在移动应用的过滤列表的基础上增加分类功能，那么使用者检索内容的效率将会更快。

在支持分类的过滤列表中，开发者可以对不同类别的列表条目进行分类。在列表内容被呈现的时候，将会显示分类类目。使用者在输入过滤条件的时候，分类类目中的文字不会被过滤筛选。如果列表条目中的内容符合过滤条件，在呈现列表条目的时候，所属分类类目也将被一同呈现。

在图9-12所呈现的界面示例中，当输入英文字母e之后，包含字母e的国家列表条目被过滤出，同时相应的分类类目也一并被显示出来。



图9-12　支持分类的过滤列表视图

要想实现支持分类功能的过滤视图，大体需要两个步骤。

首先，按照分类原则将列表条目排列在一起。例如，在上面这个示例中，我们以五大洲对国家进行分类。然后，添加各个分类类目在各类列表条目之前，例如在非洲国家列表之前添加名为Africa的分类类目，而在亚洲国家列表之前添加名为Asia的分类类目。

分类类目的添加方法为添加一个data-role属性为list-divider的列表条目。列表中的文字就是分类类目的名称，例如<li data-role="list-divider">Africa。

图9-12过滤列表视图的完整代码如代码清单9-11所示。

代码清单9-11 支持分类的过滤列表视图

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="列表视图">
      <header data-role="header">
        <h1>国家地区列表</h1>
      </header>
      <div class="content" data-role="content">
        <ul data-role="listview"  data-filter="true">
          <li data-role="list-divider">Africa</li>
          <li>Algeria</li>
          <li>Angola</li>
          <li>Benin</li>
          <li>Botswana</li>
          <li>Burkina Faso</li>
          <li data-role="list-divider">Asia</li>
          <li>Abkhazia</li>
          <li>Afghanistan</li>
          <li>Armenia</li>
          <li>Azerbaijan</li>
          <li data-role="list-divider">Europe</li>
          <li>Aland Islands</li>
          <li>Albania</li>
          <li>Andorra</li>
        </ul>
      </div>
    </section>
  </body>
</html>
```

很多情况下，列表中的条目有多种不同的表达方式，例如，刘备，姓刘，名备，字玄德，谥号昭烈皇帝，庙号烈祖，史家称为先主。因为曾经担任豫州牧，所以后人也称之为刘豫州。如此多的称呼，在列表视图中都对应到一条记录，那就是“刘备”。jQuery Mobile提供了一种隐藏数据过滤的方式，能够将所有这些信息作为索引条件进行数据过滤筛选。基于这样的方式所开发的移动应用，使用者只需要输入相应的关键字。此时当查找“刘豫州”的时候，就会得到“刘备”这个列表条目。

实现隐含数据过滤，需要列表条目上添加data-filtertext属性，并将所相关的关键词数值列在这个属性中，实现代码如下：

```
<li data-filtertext="刘备 玄德 昭烈皇帝 烈祖 刘豫州"><a href="#">刘备</a></li>
```

9.10　插页列表

插页列表是在jQuery Mobile 1.2中增加的新特性，在列表视图的外边呈现一个圆角矩形框，使用者可以很清楚地知道列表视图的范围，这样的界面呈现很适合内容相对较多的页面。在图9-13中，包含有两个列表，上方的插页列表列出了四大洋，下方的列表视图列出了主要大陆。插页列表前面的说明文字“大洋”不会被覆盖到，普通列表视图之前的说明文字被覆盖掉了，这也是普通列表视图与插页列表的不同。因为插页列表是以内联方式呈现的，所以插页列表的内容不会被覆盖掉；普通的列表视图因为不是内联方式呈现的，所以部分文字可能会被覆盖掉。



图9-13　上方为插页列表，下方为列表视图

要实现插页列表功能，需要在列表容器中声明data-inset属性为true。

图9-13所对应的页面的代码如代码清单9-12所示。

代码清单9-12　图9-13所对应的页面的代码

```
<div class="content" data-role="content">
  大洋
  <ul data-role="listview" data-inset="true">
    <li>太平洋</li>
    <li>大西洋</li>
    <li>印度洋</li>
    <li>北冰洋</li>
  </ul>
```

```
  大陆
  <ul data-role="listview">
    <li>亚欧大陆</li>
    <li>非洲大陆</li>
    <li>南美大陆</li>
    <li>北美大陆</li>
    <li>南极大陆</li>
    <li>澳大利亚大陆</li>
  </ul>
</div>
```

注意 在jQuery Mobile 1.2.0和之前版本中，所呈现的只读插页列表只是略微有点差异。在jQuery Mobile 1.2.0的只读插页列表中，列表字体、间距、列表背景颜色等发生了一些调整。经过调整之后，jQuery Mobile 1.2.0的插页列表更方便使用者阅读其中的内容。

图9-14所示的两个界面截屏分别是jQuery Mobile 1.1.0（左图）和jQuery Mobile 1.2.0的插页列表效果（右图）。我们发现，自jQuery Mobile 1.2.0之后，插页列表的高度增加了，触控操作更加方便了。



图9-14 左图为jQuery Mobile 1.1.0的只读插页列表，右图为jQuery Mobile 1.2.0的只读插页列表

因为能够通过内联方式很好地与页面内容集成在一起，并能帮助呈现出标准化的用户界面，所以插页列表是使用最为广泛的列表视图之一。

插页列表可以与几乎所有其他列表视图集成，以呈现出不同的插页列表样式，包括：

- 普通插页列表；
- 数字插页列表；

- ❑ 气泡提示插页列表；
- ❑ 缩略图插页列表；
- ❑ 分立按钮插页列表；
- ❑ 图标插页列表；
- ❑ 支持检索内容的插页列表；
- ❑ 分组插页列表（jQuery Mobile 1.2.0开始提供）。

这些列表视图在没有超级链接的时候，也都将呈现为只读插页列表的样式。

9.11　折叠列表

折叠列表视图是列表视图的一种，能够将列表折叠起来只显示列表的名称，如图9-15左图所示。如果需要，可以点击列表的名称而将折叠的列表展开，如图9-15右图所示。



图9-15　折叠列表

要实现折叠列表，需要在列表视图之外增加一个data-role为collapsible的div容器。在容器中，通过标题标签声明折叠视图的名称，示例代码如下：

```
<div data-role="collapsible">
  <h2>折叠列表标题</h2>
  <ul data-role="listview">
    <!-- 这里为列表视图的内容 -->
  </ul>
</div>
```

折叠列表中包含的列表视图可以是之前介绍的各种列表视图，例如分类列表、数字列表、分立按钮列表、缩略图和图标列表、气泡提示列表、只读列表等。

图9-15对应的代码示例如代码清单9-13所示。

代码清单9-13　折叠列表

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="列表视图演示">
      <header data-role="header">
        <h1>折叠列表视图</h1>
      </header>
      <div data-role="collapsible">
        <h2>省份列表</h2>
        <ul data-role="listview">
          <li>北京</li>
          <li>天津</li>
          <li>河北</li>
          <li>河南</li>
          <li>山西</li>
          <li>山东</li>
          <li>内蒙古</li>
          <li>湖北</li>
          <li>湖南</li>
          <li>广东</li>
          <li>广西</li>
        </ul>
      </div>
    </section>
  </body>
</html>
```


如果需要将多个折叠列表视图以集合的形式排列在一起，可以使用折叠列表集合。如果这个集合的各个列表都折叠起来，那么这个折叠列表集合的呈现效果就好像各个列表标题又组成一个列表。打开每个折叠列表的标题之后，又会呈现其中的视图内容，效果如图9-16所示。

要实现折叠列表集合，首先要建立data-role属性为collapsible-set的div容器。在这个div容器中，顺序排列了各个折叠列表视图。这样，各个折叠列表集成在一个更大的集合中，就形成了折叠列表集合，其代码如下所示：

```
<div data-role="collapsible-set">
  <div data-role="collapsible">
    <h2>第一个折叠列表</h2>
    <ul data-role="listview">
      <!-- 这里为列表视图内容 -->
    </ul>
  </div>
  <div data-role="collapsible">
```

```
      <h2>第二个折叠列表</h2>
      <ul data-role="listview">
        <!-- 这里为列表视图内容 -->
      </ul>
    </div>
  </div>
```



图9-16　折叠列表集合

在折叠列表集合中，各个折叠列表视图默认是折叠起来的。每次只能展开一个折叠列表视图，当展开第二个时，前一个会自动折叠起来。

图9-16对应的代码如代码清单9-14所示。

代码清单9-14　折叠列表集合

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="列表视图演示">
      <header data-role="header">
        <h1>折叠列表集合</h1>
      </header>
      <div data-role="collapsible-set" data-theme="b" data-content-theme="d">
        <div data-role="collapsible">
          <h2>华北</h2>
```

```
        <ul data-role="listview">
          <li>北京</li>
          <li>天津</li>
          <li>河北</li>
          <li>山西</li>
          <li>内蒙古西部</li>
        </ul>
      </div>
      <div data-role="collapsible">
        <h2>东北</h2>
        <ul data-role="listview">
          <li>辽宁</li>
          <li>吉林</li>
          <li>黑龙江</li>
          <li>内蒙古东部</li>
        </ul>
      </div>
      <div data-role="collapsible">
        <h2>华中</h2>
        <ul data-role="listview">
          <li>河南</li>
          <li>湖北</li>
          <li>湖南</li>
        </ul>
      </div>
    </div>
  </section>
 </body>
</html>
```

默认的折叠列表集合都是以内联样式呈现的，除非特别声明列表视图是非内联的。声明折叠列表集合为非内联方式时，需要在折叠列表集合的容器中将data-inset属性设置为false：

```
<div data-role="collapsible-set" data-inset="false">
  <!-- 这里是各个集合列表视图-->
</div>
```

小经验 折叠列表和折叠内容块从HTML的属性定义到呈现样式上都非常接近，折叠列表或折叠列表集合是面向列表视图设计的，而折叠内容块或折叠内容集合是面向文本、图片等内容设计的。

这里需要说明的是，在jQuery Mobile 1.2.0之前的版本中并不支持折叠列表。

9.12 自动分类列表视图

在之前的分类列表中，我们通过程序设定分类标签并由此进行分类。从1.2.0版本开始，jQuery Mobile提供了能够自动分类的列表视图。基于自动分类列表视图，对于列表条目相邻的内容，如果第一个字符或第一个汉字相同，那么将会被自动分类在一起，而分类标签就是第一个字符或者

第一个汉字。这样的功能设计有助于用户快速识别与定位所要查找的内容，如果与检索列表内容的功能配合使用，或者将折叠列表与自动分类列表视图混合使用，将可能明显改善列表内容查找的用户体验。图9-17就是这样的自动分类列表视图。



图9-17　自动分类列表视图

要实现自动分类列表视图，大致需要两个步骤。

(1) 在列表内容输出的时候，需要排序之后再输出到列表视图中，否则，即便存在两个列表条目，它们的首字母或第一个汉字相同但是不相邻在一起，也是无法实现自动分类之后在一起呈现的效果的。

(2) 在列表视图DOM容器中，声明data-autodividers属性为true：

```
<ul data-role="listview" data-autodividers="true">
```

完成这样两个步骤之后，就可以呈现自动列表视图了。

完整的自动列表视图的代码如代码清单9-15所示。

代码清单9-15　自动列表视图

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="列表视图演示">
```

```
        <header data-role="header">
          <h1>自动分类列表视图</h1>
        </header>
        <div class="content" data-role="content">
          <ul data-role="listview" data-dividertheme="e" data-autodividers="true">
            <li>北京</li>
            <li>天津</li>
            <li>河北</li>
            <li>河南</li>
            <li>山西</li>
            <li>山东</li>
            <li>内蒙古</li>
            <li>湖北</li>
            <li>湖南</li>
            <li>广东</li>
            <li>广西</li>
          </ul>
        </div>
      </section>
    </body>
</html>
```

自动分类列表视图和之前介绍的列表视图有两个不同之处，具体如下所示。

- 在DOM容器中，自动分类列表视图增加了`data-autodividers`属性，这在列表视图中是没有的。
- 在列表视图中，可以在列表项目中通过设定`data-role`属性为`list-divider`来标记这个条目是分类标签，这在自动分类列表视图中是不需要的。

自动分类列表也可以与前一节的折叠列表混合使用，以方便用户快速定位内容，如图9-18所示。在这个省份列表的场景中，折叠列表中嵌套的是一个自动分类列表视图的省份列表。



图9-18 折叠列表与自动分类列表混合使用的场景

9

在图9-18所示的应用场景中，折叠列表的div容器被定义在外层，自动分类列表的ul容器被定义在内层，对应的完整代码如代码清单9-16所示。

代码清单9-16　折叠列表与自动分类列表视图混合使用

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="列表视图演示">
    <header data-role="header">
      <h1>自动分类列表的混合使用</h1>
    </header>
    <div data-role="collapsible">
      <h2>省份列表</h2>
        <ul data-role="listview" data-dividertheme="e" data-autodividers="true">
          <li>北京</li>
          <li>天津</li>
          <li>河北</li>
          <li>河南</li>
          <li>山西</li>
          <li>山东</li>
          <li>内蒙古</li>
          <li>湖北</li>
          <li>湖南</li>
          <li>广东</li>
          <li>广西</li>
        </ul>
      </div>
    </section>
  </body>
</html>
```

注意　开发者在生成自动分类列表视图的列表内容时，最好根据首字母或者第一个汉字进行排序。如果列表内容不按照首字母或者第一个汉字排序，可能会出现两个首字母或第一个汉字名称相同的分组，且位于自动分类列表的不同位置。

9.13　使用列表美化表单布局

列表或者插页列表可以用于美化表单样式。经过美化之后，表单元素的布局更加规整，表单

操作起来更加方便，如图9-19所示。



图9-19　包含有表单的插页列表

有很多方法可以使用列表视图来美化表单。在这些列表视图样式中，使用最多的是插页列表或者只读列表视图。对于表单元素的排列方法，一般情况下是每行一个表单元素。

使用列表格式化表单布局时，在将表单元素依次排列在列表视图之外，还有两个细节需要留意。

(1) 作为表单元素的`<li>`容器需要设置`data-role`属性为`fieldcontain`。

(2) 如果将按钮作为列表的一部分，则需要将其放入`fieldset`容器，然后再将`fieldset`放入`<li>`容器中。

经过列表视图美化的图9-19的登录界面的代码可参见代码清单9-17。

9

代码清单9-17　使用列表视图美化的登录界面的代码

```
<div class="content" data-role="content">
  登录
  <form>
    <ul data-role="listview" data-inset="true">
      <li data-role="fieldcontain">
        <label for="name">登录名:</label>
        <input type="text" name="name" id="name" value=""  />
      </li>
      <li data-role="fieldcontain">
        <label for="name">密码:</label>
        <input type="Password" name="password" id="password" value=""/>
      </li>
      <li data-role="fieldcontain">
        <fieldset class="ui-grid-a">
          <div class="ui-block-a">
```

```
        <button type="submit" data-theme="d">取消</button>
      </div>
      <div class="ui-block-b">
        <button type="submit" data-theme="a">提交</button>
      </div>
    </fieldset>
  </li>
</ul>
</form>
</div>
```

在上面的代码中，我们并没有使用内联按钮样式进行布局，而是通过声明布局的样式进行布局管理。首先，在fieldset中通过ui-grid-a声明每行包含两栏，然后在各个按钮中声明ui-block-a为第一栏内容，ui-block-b为第二栏内容，具体内容请参考11.3节。

9.14　美化列表内容

在列表视图中使用HTML和CSS来美化排版样式，将可以设计出很多有趣的列表页面效果。在图9-20所示的界面中，分别集成了下面这些技术：

- 分类列表视图；
- 列表标题；
- 列表内容；
- 超级链接的列表项目与图标显示；
- 列表条目右上角的备注信息，一般为“了解更多”。



图9-20　列表视图的美化

要实现这种经过HTML排版的界面样式，首先要在各个列表条目内部进行HTML排版。下面这段代码实现了一个典型的用HTML排版的列表：

```
<li>
  <a href="#">
    <h1>印度洋</h1>
    <p>世界第三大洋，7491万平方公里</p>
    <p class="ui-li-aside">了解更多</p>
  </a>
</li>
```

在经过HTML排版美化的列表视图内，列表条目标题通过<h1>到<h6>的标题实现。至少在已经正式发行的jQuery Mobile版本中，<h1>到<h6>在这样的列表视图中所呈现的效果是一样的。

在标题之后，通过<p>分段标签包含各个说明文字，例如在图9-20中<p>标签包含了对各个地理概念的介绍。在<p>段落标签中，设定class属性为ui-li-aside，表示这段文字位于列表视图条目的右上方。

图9-20所示页面的完整代码如代码清单9-18所示。

代码清单9-18　经过HTML排版美化的列表视图

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="列表视图演示">
      <header data-role="header">
        <h1>列表视图样式美化</h1>
      </header>
      <div class="content" data-role="content">
        <ul data-role="listview" data-dividertheme="e">
          <li data-role="list-divider">大洋</li>
          <li><a href="#">
            <h1>
            <h1>太平洋</h1>
            <p>面积最大，1亿5千多万平方公里</p>
            <p class="ui-li-aside">了解更多</p>
            </a></li>
          <li><a href="#">
            <h1>大西洋</h1>
            <p>世界第二大洋，8221万平方公里</p>
            <p class="ui-li-aside">了解更多</p>
          </a></li>
          <li><a href="#">
            <h1>印度洋</h1>
            <p>世界第三大洋，7491万平方公里</p>
            <p class="ui-li-aside">了解更多</p>
          </a></li>
          <li><a href="#">
```

```
        <h1>北冰洋</h1>
        <p>1405万平方公里</p>
        <p class="ui-li-aside">了解更多</p>
      </a></li>
      <li data-role="list-divider">州别</li>
      <li><a href="#">
        <h1>亚洲</h1>
        <p>4382万平方公里</p>
        <p class="ui-li-aside">了解更多</p>
      </a></li>
      <li><a href="#">
        <h1>非洲</h1>
        <p>3037万平方公里</p>
        <p class="ui-li-aside">了解更多</p>
      </a></li>
      <li><a href="#">
        <h1>北美洲</h1>
        <p>2449万平方公里</p>
        <p class="ui-li-aside">了解更多</p>
      </a></li>
      <li><a href="#">
        <h1>南美洲</h1>
        <p>1784万平方公里</p>
        <p class="ui-li-aside">了解更多</p>
      </a></li>
      <li><a href="#">
        <h1>南极洲</h1>
        <p>1372万平方公里</p>
        <p class="ui-li-aside">了解更多</p>
      </a></li>
      <li><a href="#">
        <h1>欧洲</h1>
        <p>1018万平方公里</p>
        <p class="ui-li-aside">了解更多</p>
      </a></li>
      <li><a href="#">
        <h1>大洋洲</h1>
        <p>900万平方公里</p>
        <p class="ui-li-aside">了解更多</p>
      </a></li>
    </ul>
  </div>
 </section>
</body>
</html>
```

由于分段标签<p>所包含的内容一般不会折行显示，所以其中的文字长度通常会受到限制。如果分段标签<p>内的文字较多，jQuery Mobile将会隐藏超过宽度的文字，这将影响用户的阅读体验。例如，单独修改印度洋的介绍，加入更多的相关信息：

```
<li>
  <a href="#">
    <h1>印度洋</h1>
```

```
    <p>世界第三大洋，7491万平方公里</p>
    <p>位于亚洲、非洲、大洋洲和南极洲之间，大部分在南半球，约占世界海洋总面积的21.1%。印度洋的范围
     北至印度次大陆及阿拉伯半岛（南亚及西亚），西达东非，东侧则以印度尼西亚、巽他群岛及澳大利亚为界，
     南迄南冰洋（也有定义至南极洲）。</p>
    <p class="ui-li-aside">了解更多</p>
  </a>
</li>
```

从图9-21中可以看到，如果文字超过屏幕显示范围，则超过部分会被覆盖掉，所以建议开发者在排版列表视图的内容时，注意一下字数。鉴于不同移动设备的显示屏尺寸不同，所能容纳的文字长度也不同，所以目标受众的移动设备显示屏尺寸也是在移动应用设计阶段需要考虑的。



图9-21 文字超长的说明文字

如果一定要在列表条目中包含大量问题，而且支持折行显示的话，可以对<p>段落标签定义单独的CSS样式，声明以正常方式显示，相关代码如下：

```
<style type="text/css">
  .longContent {
    white-space: normal !important;
  }
</style>
```

注意，需要在这个CSS定义中声明`!important`，以保证这个设置能够生效。

然后，在相应的<p>段落标签中设置这个CSS样式定义：

```
<p class="longContent">…</p>
```

说明文字超长的列表视图的完整代码如代码清单9-19所示，呈现效果如图9-22所示。

代码清单9-19 支持折行显示的列表视图

```
<!DOCTYPE html>
<html>
  <head>
```

9

```
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <style type="text/css">
      .longContent {
        white-space: normal !important;
      }
    </style>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="列表视图演示">
      <header data-role="header">
        <h1>列表视图样式美化</h1>
      </header>
      <div class="content" data-role="content">
        <ul data-role="listview" data-dividertheme="e">
          <li data-role="list-divider">大洋</li>
          <li><a href="#">
            <h1>印度洋</h1>
            <p>世界第三大洋，7491万平方公里</p>
            <p class="longContent">位于亚洲、非洲、大洋洲和南极洲之间，大部分在南半球。约占世界
              海洋总面积的21.1%。印度洋的范围北至印度次大陆及阿拉伯半岛（南亚及西亚），西达东非，
              东侧则以印度尼西亚、巽他群岛及澳大利亚为界，南迄南冰洋（也有定义至南极洲）。</p>
            <p class="ui-li-aside">了解更多</p>
          </a></li>
        </ul>
      </div>
    </section>
  </body>
</html>
```



图9-22　折行显示效果

9.15　列表视图属性、选项、方法和事件

在开发列表视图功能的时候，可以通过JavaScript对其界面风格样式与主题进行设定，也可以通过列表视图方法刷新或获得下一级页面的内容。

9.15.1　属性

列表视图属性是定义在列表视图容器中的属性，用以定义列表视图的不同页面元素样式。常用的列表视图属性如表9-1所示。

表9-1　列表视图属性

属　性	含　义
data-count-theme	设置气泡提示主题风格，默认为c。 示例代码： <ul data-role="listview" data-count-theme="e">
data-dividertheme	设置分类列表中分类栏目的显示主题风格，默认为b。 此属性只对分类列表有效，对其他类型列表无效。 示例代码： <ul data-role="listview" data-dividertheme="e">
data-filter	设置是否显示检索工具栏。 默认为false，不显示检索工具栏。 设置为true，则显示检索工具栏。 示例代码： <ul data-role="listview" data-filter="true">
data-filter-placeholder	设置检索工具栏的背景文字。默认为Filter items…。 此属性只对具有检索功能的列表有效。如果列表没有检索功能，则此属性无效。 示例代码： <ul data-role="listview" data-filter="true" data-filter-placeholder="查找">
data-filter-theme	设置检索工具栏的主题样式。默认为b主题样式。 此属性只对具有检索功能的列表有效。如果列表没有检索功能，则此属性无效。 示例代码： <ul data-role="listview" data-filter="true" data-filter-theme="e">
data-header-theme	设置嵌套列表表头的主题样式风格，默认为b风格。 示例代码： <ul data-role="listview" data-filter="true" data-header-theme="e">
data-inset	设置为插页风格。 默认为false，不使用插页风格。 设置为true，使用插页风格样式。 示例代码： <ul data-role="listview" data-inset="true">

（续）

属　　性	含　　义
data-split-icon	设置分立按钮图标样式。经过设置，分立图标按钮在没有设置图标样式的情况下，默认使用被设置的图标样式。这里也可以使用自定义图标样式。 默认为向右箭头arrow-r。 示例代码： `<ul data-role="listview" data-split-icon="plus">`
data-split-theme	设置分立按钮的列表主题样式，默认为b。 示例代码： `<ul data-role="listview" data-split-theme="e">`
data-theme	设置列表主题样式。默认情况下，主题样式继承上一级容器的主题样式。 示例代码： `<ul data-role="listview" data-theme="e">`

9.15.2　选项

在jQuery Mobile中，列表的选项与属性的功能大致相当。选项通常在JavaScript中的初始化中用到，例如在mobileinit事件中进行选项设定，则可以对相应页面中各个列表视图进行统一的样式和行为设定。

由于列表选项与属性大多类似，就不对类似的内容重复介绍了。表9-2是列表选项与属性的对照表。

表9-2　列表选项与属性的对照表

选　　项	属　　性	功　　能
countTheme	data-count-theme	设置气泡提示主题风格
dividerTheme	data-dividetheme	设置分类列表中分类栏目的显示主题风格
filter	data-filter	设置是否显示检索工具栏
filterPlaceholder	data-filter-placeholder	设置检索工具栏的背景文字
filterTheme	data-filter-theme	设置检索工具栏的主题样式
headerTheme	data-header-theme	设置嵌套列表表头的主题样式风格
inset	data-inset	设置为插页风格
splitIcon	data-split-icon	设置分立按钮图标样式
splitTheme	data-split-theme	设置分立按钮列表主题样式
theme	data-theme	设置列表主题样式

在jQuery Mobile中，有两个选项没有对应的属性，具体如下所示。

- initSelector：用于对特定CSS选择器指定的按钮进行美化，其默认值为jqmData(role='listview')。开发者可以设定特定的CSS样式名称，设定之后，jQuery Mobile将会对其进

行列表样式美化。而在这样的页面下，data-role="listview"也不再起作用。

- filterCallback：用以设定回调函数来增强jQuery Mobile检索列表的实现功能。

9.15.3 方法和事件

列表视图提供了两个方法来获得子页或者刷新列表视图样式，具体如表9-3所示。

表9-3 列表视图的方法

方　　法	功　　能
childPages	获得嵌套列表子页的对象
refresh	刷新列表视图样式

此外，需要说明的是，在创建列表视图时，将会触发create事件。

9.16 高级编程技巧

在进行列表视图开发的时候，一些高级技巧将会有助于开发用户体验更好的应用。在这一节中，我们将会介绍：

- 移除列表视图各个条目之间分隔线的方法；
- 列表视图支持触控操作的方法；
- 动态加载列表视图内容。

9.16.1 移除各列表条目间的分隔线

当使用列表或者插页列表方式美化表单时，表单元素之间总有一个分隔线。而在通常的表单操作中，这样的分隔线并不美观，如图9-23左图所示。一种解决方法是通过移除ui-body-c样式来移除列表视图的表格线，详情可参见代码清单9-20，得到的最终效果如图9-23右图所示。

9

代码清单9-20 移除表单中ui-body-c的CSS样式定义

```
<script type="text/javascript">
  $(document).ready(function(e) {
    $('#listViewForm').children().removeClass('ui-body-c');
  });
</script>
```

代码清单9-21是通过列表方式美化表单的HTML代码片段。注意，在这个代码片段中，我们声明了列表视图的DOM容器的id属性为listViewForm，而相应的JavaScript也是对这个id为listViewForm的对象移除了ui-body-c的CSS定义。



图9-23　左图为通常的表单，右侧为去除分割线的美化效果

代码清单9-21　去除列表视图分隔线的表单

```
<form>
  <ul data-role="listview" data-inset="true" id="listViewForm">
    <li data-role="fieldcontain">
      <label for="name">登录名:</label>
      <input type="text" name="name" id="name" value=""  />
    </li>
    <li data-role="fieldcontain">
      <label for="name">密码:</label>
      <input type="Password" name="password" id="password" value=""  />
    </li>
    <li data-role="fieldcontain">
      <fieldset class="ui-grid-a">
        <div class="ui-block-a">
          <button type="submit" data-theme="d">取消</button>
        </div>
        <div class="ui-block-b">
          <button type="submit" data-theme="a">提交</button>
        </div>
      </fieldset>
    </li>
  </ul>
</form>
```

与之前“使用列表美化表单样式”略有区别，在这样的应用场景中必须对`<ul>`容器设置id，以便jQuery选择器去除相应的CSS样式。

9.16.2　列表视图触控操作

在第6章中，我们介绍了对页面进行触控操作的实现方法。在列表视图中，我们同样可以通

过对列表进行触控操作以实现更加丰富的功能。对于很多使用列表视图的场景，触控操作将会使列表视图更加方便。例如，在iPhone手机中，如果使用嵌套列表进行内容管理与呈现，那么从下级列表返回上级列表会很难操作，毕竟iPhone手机没有Android那样的返回键。如果能在列表视图中支持触控操作，就可以通过轻扫屏幕实现从下级列表视图返回到上一级列表视图了。

在图9-24中，当向左轻扫列表内容时，则显示关于对话框。而点击列表内容时，则会跳转到相应的页面中。

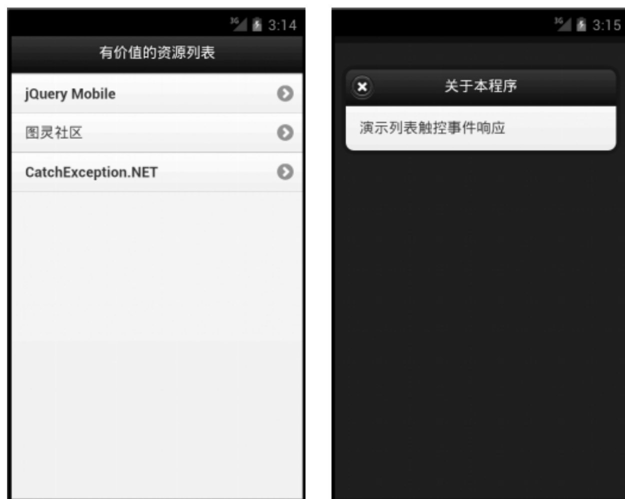


图9-24　左图为初始的界面，向左轻扫后显示右图所示的对话框

实现图9-24这样的触控操作，需要将触控事件响应函数绑定在相应列表项上。代码清单9-22是触控事件绑定的JavaScript代码片段。

代码清单9-22　基于列表视图的触控事件响应

```
<script type="text/javascript">
  $('#listitem_jQueryMobile').live('swipeleft',function(){
    $('#lnkDialog').click();
  });

  $('#listitem_ituring').live('swipeleft',function(){
    $('#lnkDialog').click();
  });

  $('#listitem_CatchException').live('swipeleft',function(){
    $('#lnkDialog').click();
  });
</script>
```

为了能够打开多页模板中的对话框界面，这里使用一个隐藏的超级链接以帮助实现点击操作，如代码清单9-23所示。

代码清单9-23　用以打开对话框的隐藏的超级链接

```
<a id='lnkDialog' href="#about" data-rel="dialog" data-transition="pop" style='display:none;'></a>
```

如果跳转目标不是当前HTML文件中的某个页面而是一个外部网页，则不需要使用这个技巧。

下面是对应的列表部分的代码片段，当特定列表项被向左扫过时，绑定其上的触控事件被触发，此时会打开对话框页面：

```
<ul data-role="listview">
  <li id="listitem_jQueryMobile">
    <a href="http://jquerymobile.com" rel="external">jQuery Mobile</a>
  </li>
  <li id="listitem_ituring">
    <a href="http://www.ituring.com.cn" rel="external">图灵社区</a>
  </li>
  <li id="listitem_CatchException">
    <a href="http://catchexception.net" rel="external">CatchException.NET</a>
  </li>
</ul>
```

9.16.3　动态加载列表视图

由于列表视图中列表的DOM内容是一次性加载和呈现的，如果页面中包含比较多的列表项，则下载和加载速度都会比较慢，此时建议使用延迟加载技术，即打开界面时只有一部分列表内容被呈现，随着页面向下翻动，后续的内容被继续加载。这样，将一次加载所有内容变为每次加载一部分内容，那么每次的加载和显示速度就会更快。另一个好处在于，如果面对大量公众用户的使用场景，这样的策略也可以减少服务器的压力。系统不再需要一次性将所有数据传递给移动设备，而是随用随取，用户需要浏览多少内容，就下载多少内容。

这个功能的实现主要通过以下两个部分：

- 检测页面是否滚动到底部，如果是，则执行延迟加载；
- 通过延迟加载函数将后续列表的内容加载到列表中。

还要注意一点，由于这个延迟加载程序通过监测页面是否滚动到底部而触发加载活动，所以第一屏的列表内容需要能保证排列之后超过一屏尺寸，否则无法执行屏幕滚动操作，也就无法触发延迟加载操作。在实际的项目实践中，建议在列表下方增加一个“加载更多”按钮，在自动加载失败的时候，使用者可以手工触发这个按钮实现页面剩余内容的加载，例如微博或Facebook都有类似的设计。

代码清单9-24用于检测界面是否滚动到屏幕底部，并执行加载活动的代码片段。

代码清单9-24　动态加载HTML DOM内容

```
<script type="text/javascript">
  /* 检测页面是否滚动到最低部 */
  $(window).scroll(function(){
    if($(window).scrollTop()==$(document).height()-$(window).height()){
      loadMore();
```

```
    }
  });

  /* 延迟加载函数 */
  function loadMore(){
    var i=0;
    for(i=0; i<5; i++)
    {
      /* 根据业务场景的不同而定义不同的加载内容 */
      var j = 0;
      var newListItem = document.createElement('li');
      newListItem.innerHTML = 'List Item';
      $('#listviewScrolling').append(newListItem);
    }

    $('#listviewScrolling').listview('refresh');
  }
</script>
```

上面这个程序会动态创建列表项内容并将其添加在列表最后。这个示例的完整代码如代码清单9-25所示。

代码清单9-25 动态加载列表视图

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type="text/javascript">
      /* 检测页面是否滚动到最低部 */
      $(window).scroll(function(){
        if($(window).scrollTop()==$(document).height()-$(window).height()){
          loadMore();
        }
      });

      /* 延迟加载函数 */
      function loadMore(){
        var i=0;
        for(i=0; i<5; i++)
        {
          /* 根据业务场景的不同定义不同的加载内容 */
          var j = 0;
          var newListItem = document.createElement('li');
          newListItem.innerHTML = 'List Item';
          $('#listviewScrolling').append(newListItem);
        }

        $('#listviewScrolling').listview('refresh');
```

```
      }
    </script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="列表视图演示">
      <header data-role="header">
        <h1>延迟加载</h1>
      </header>
      <div class="content" data-role="content">
        <ul data-role="listview" id="listviewScrolling">
          <li>List Item</li>
          <li>List Item</li>
          <li>List Item</li>
          <li>List Item</li>
          <li>List Item</li>
          <li>List Item</li>
          <li>List Item</li>
          <li>List Item</li>
          <li>List Item</li>
          <li>List Item</li>
          <li>List Item</li>
          <li>List Item</li>
          <li>List Item</li>
          <li>List Item</li>
          <li>List Item</li>
        </ul>
      </div>
    </section>
  </body>
</html>
```

在实际应用场景中，列表内容可能需要动态取自服务器，所以一个完整的动态列表视图加载往往是浏览器和服务器共同完成的。

在移动设备浏览器每次执行加载的时候，由浏览器以Ajax方式向服务器发起请求以获得后续的内容。服务器将生成的内容以JSON的方式传回浏览器。浏览器中的JavaScript在接收到来自服务器的内容之后，对内容进行解析并生成合适的DOM对象，然后将新的列表条目添加在列表视图后面。这样就是一个基本的服务器与浏览器共同完成的动态加载过程。

由于这里主要介绍jQuery Mobile移动应用开发技术，所以Ajax通信以及服务器处理相关的内容就不赘述了。

第 10 章

表　　单

10

表单是HTML中最常用的技术之一，通常用于向服务器提交输入的内容。在jQuery Mobile中，表单所实现的功能和以往的HTML表单完全一样，但是经过美化的界面更加简洁，实现起来也更加简单。

在这一章中，我们将了解以下内容：

- 表单样式；
- 输入框；
- 单选按钮；
- 复选框；
- 滑块；
- 开关按钮；
- 选择菜单；
- 禁用表单元素；
- 隐藏标签；
- Mini尺寸的表单样式；
- 高级开发技术。

10.1　表单样式

jQuery Mobile表单与其他HTML表单的提交方式一致，也是通过基于<form>标签的post或者get方法实现的。提交表单的原型化代码如下：

```
<form id="FormID" name="FormName" method="post" action="index.php">...</form>
```

默认情况下，表单元素的样式将会继承页面或者表单的样式。通常，开发者不需要专门美化表单样式，而是由jQuery Mobile自动完成。图10-1是一个登录表单，这个表单中的各个元素经过jQuery Mobile美化之后，将呈现与其他页面元素风格一致的样式，比如输入框和按钮等。



图10-1　登录表单

图10-1所示的登录表单的代码如代码清单10-1所示。

代码清单10-1　登录表单

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="列表视图演示">
      <header data-role="header">
        <h1>登录</h1>
      </header>
      <div class="content" data-role="content">
        <form>
          <div data-role="fieldcontain">
            <input type="text" name="name" id="name" placeholder="登录名" />
            <input type="Password" name="password" id="password" placeholder="密码"/>
            <div style="margin-top:20px;" >
              <fieldset class="ui-grid-a">
                <div class="ui-block-a">
                  <button type="reset" data-theme="d">取消</button>
                </div>
                <div class="ui-block-b">
                  <button type="submit" data-theme="a">提交</button>
                </div>
              </fieldset>
            </div>
```

```
          </div>
        </form>
      </div>
    </section>
  </body>
</html>
```

如果希望表单以原生的HTML表单样式呈现，需要进行一些额外的设置。通过修改data-role属性为none，表单元素将以原生表单样式而非移动应用风格样式呈现。例如，在代码清单10-1中，在文本输入框的相关代码中加入data-role="none"，那么文本输入框就以原生表单样式显示，如图10-2所示。



图10-2　原生表单样式

10.2　输入框

按照功能，输入框被细分为如下14种，具体如表10-1所示。

10

表10-1　输入框类型

类　　型	名　　称
text	文本输入框
password	密码输入框
number	数字输入框
email	电子邮件输入框
url	URL地址输入框
tel	电话号码输入框

（续）

类　　型	名　　称
time	时间输入框
data	日期输入框
month	月份输入框
week	周输入框
datetime	时间日期输入框
datetime-local	本地时间日期输入框
color	色彩输入框
search	查询框

每种输入框均可以通过这样的形式在jQuery Mobile中使用：

```
<input type="text" name="name" id="name" value=""/>
```

例如电话号码输入框在jQuery Mobile中是这样使用的：

```
<input type="tel" name="name" id="name" value=""/>
```

各个输入框的type属性可能不同，但是它们的呈现样式是一致的。

10.2.1　属性与选项

输入框的属性与选项的实现功能大致相同，重复的部分这里就不再赘述了。使用这些属性和选项，我们可以设定输入框的尺寸、缩放控制以及主题风格。表10-2列出了输入框的属性、选项及其功能。

表10-2　输入框属性与选项

选　　项	属　　性	功　　能
mini	data-mini	设置标准尺寸或者mini尺寸，默认值为false
preventFocusZoom	data-prevent-focus-zoom	当焦点位于输入框中时，禁止执行缩放操作，以免缩放后输入困难。在iOS平台中，默认值为true，表示禁止缩放
theme	data-theme	设置主题风格。默认为空，表示主题风格继承自父级容器

下面举例说明如何通过属性和选项来设置输入框的主题风格。通过属性设置时间输入框主题风格的代码如下：

```
<input type="time" id="myTimeInput" data-theme="e"/>
```

通过选项设置时间输入框主题风格的代码如下：

```
$('# myTimeInput).slider({ theme: "e" });
```

除了表10-2提到的3个选项外，输入框还有一个选项，那就是initSelector选项，它用以设定

自定义的选择器。如果设定了initSelector，那么只有选择器指定的输入框才会被jQuery Mobile进行样式渲染和通过选项、属性、方法等进行控制。

10.2.2　方法与事件

在文本输入框中，主要涉及两个方法，分别是enable()和disable()，其中前者用于启用一个已禁用的输入框，后者用于禁用一个输入框。下面这行代码实现了对某个输入框的禁用操作：

```
$('#myInput').textinput('disable');
```

输入框支持创建事件。在创建输入框时，会触发create事件。

10.3　单选按钮

单选按钮可以包含若干个多选一的选项，使用者每次只能选择一个选项。被选中的选项将会随表单而一同被提交到服务器。jQuery Mobile的单选按钮样式如图10-3所示。在jQuery Mobile中，没有选中的单选按钮是灰色的，而选中的单选按钮则会高亮显示。不管选中或者没有选中，按钮的文字都不会发生变化。



图10-3　单选按钮

在移动应用中，为方便用户作出选择，单选按钮通常以按钮组的形式呈现。要实现按钮组，开发者需要将各个单选按钮置于fieldset容器中，并将该容器的data-role属性设置为controlgroup。通常，一个fieldset只作为一个按钮组使用。如果有多组不同的单选按钮，则可以在不同的fieldset容器中分别放置各组单选按钮。

代码清单10-2是图10-3所示页面的代码。

代码清单10-2　单选按钮

```
<div data-role="fieldcontain">
  <fieldset data-role="controlgroup">
    <legend>单选按钮：</legend>
    <input type="radio" name="radio-choice-1" value="choice-1" checked="checked"/>
    <label for="radio-choice-1">C++</label>
    <input type="radio" name="radio-choice-1" value="choice-2"/>
    <label for="radio-choice-2">Pascal</label>
    <input type="radio" name="radio-choice-1" value="choice-3"/>
    <label for="radio-choice-3">Visual C++</label>
    <input type="radio" name="radio-choice-1" value="choice-4"/>
```

10

```
    <label for="radio-choice-4">Delphi</label>
    <input type="radio" name="radio-choice-1" value="choice-5"/>
    <label for="radio-choice-5">JavaScript</label>
  </fieldset>
</div>
```

默认情况下，单选按钮是自上而下依次排列的。如果想水平排列单选按钮，则需要在<filedset>容器中声明data-type属性为horizontal。需要注意的是，水平排列会受浏览器分辨率的影响。如果一行文字较多，可能出现换行，从而影响呈现效果。水平排列的单选按钮的代码结构如代码清单10-3所示。

代码清单10-3　水平排列的单选按钮

```
<div data-role="fieldcontain">
  <fieldset data-role="controlgroup" data-type="horizontal">
    <legend>水平单选按钮：</legend>
    <input type="radio" name="radio-choice-b" value="001" checked="checked" />
    <label for="radio-choice-c">C++</label>
    <input type="radio" name="radio-choice-b" value="010" />
    <label for="radio-choice-d">Pascal</label>
    <input type="radio" name="radio-choice-b" value="100" />
    <label for="radio-choice-e">Python</label>
  </fieldset>
</div>
```

运行上述代码，得到的结果如图10-4所示。



图10-4　水平排列的单选按钮

在开发与单选按钮相关的功能中，开发者可以通过jQuery Mobile所提供的属性与选项对单选按钮的尺寸、主题风格等进行设定，通过方法和事件对单选按钮的行为进行处理。

10.3.1　属性与选项

单选按钮具有两个选项，分别用于设定单选按钮的尺寸和主题风格。表10-3列出了单选按钮属性、选项及其功能的对应关系。

表10-3　单选按钮的属性与选项

选　　项	属　　性	功　　能
mini	data-mini	设置标准尺寸，或者mini尺寸，默认为false
theme	data-theme	设置主题风格。默认为空，主题风格继承自父级容器

下面我们看一下如何通过属性或者选项来实现主题风格。通过属性设置主题风格的代码如下：

```
<input type="radio" name="myRadio" id="myRadio" value="001" data-theme="e" />
```

通过选项设置主题风格的代码如下：

```
$('#myRadio').checkboxradio({ theme: 'e' });
```

10.3.2 方法与事件

在单选按钮的相关操作中，主要涉及3个方法，分别是enable()、disable()和refresh()，其作用分别是启用已禁用的单选按钮，禁用某个单选按钮，刷新单选按钮。禁用单选按钮的示例代码如下：

```
$('#myRadio').checkboxradio('disable');
```

单选按钮支持创建事件。在创建单选按钮时，会触发create事件。

10.4 复选框

复选框允许用户一次性选择多个选项，如图10-5所示。有别于单选按钮，复选框可以支持同时选择多个不同的选项。被选中的复选框会高亮显示包含有对钩的矩形框，没有选中的复选框则会呈现为灰色的矩形框。

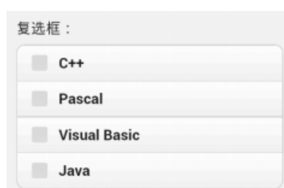


图10-5 复选框

实现图10-5所示的复选框的功能的代码如代码清单10-4所示。同单选按钮一样，复选框通常也会被放置在fieldset容器中，并且该容器的data-role属性被设置为controlgroup，表示它们为一组相关的复选框。

代码清单10-4 复选框

```
<div data-role="fieldcontain">
  <fieldset data-role="controlgroup">
    <legend>复选框：</legend>
    <input type="checkbox" name="checkbox-1a" id="checkbox-1a" />
    <label for="checkbox-1a">C++</label>
    <input type="checkbox" name="checkbox-2a" id="checkbox-2a"  />
    <label for="checkbox-2a">Pascal</label>
    <input type="checkbox" name="checkbox-3a" id="checkbox-3a" c />
    <label for="checkbox-3a">Visual Basic</label>
    <input type="checkbox" name="checkbox-4a" id="checkbox-4a" />
    <label for="checkbox-4a">Java</label>
```

```
  </fieldset>
</div>
```

如果单独来看单选按钮或者复选框的HTML代码，我们会发现这样的代码和标准的HTML Form代码没有任何差别。而它们的呈现效果的确是jQuery Mobile的Web移动应用的风格。这是由于默认情况下，所有页面中的单选按钮或复选框都会被jQuery Mobile渲染成统一的风格样式。

同单选按钮一样，在复选框的fieldset容器中声明data-type属性为horizontal，可以实现复选框的水平布局。需要注意的是，水平排列也会受到浏览器分辨率的影响，文字过多将可能出现折行，从而影响最终呈现效果。因此，开发者需要谨慎选择是否使用水平布局呈现复选框。

虽然复选框和单选按钮在功能和样式上存在诸多不同，但是各个属性、选项、方法与事件的定义是完全一样的。比如，下面的代码用于禁用复选框：

```
$('#myCheckbox').checkboxradio('disable');
```

与单选按钮非常类似，除了jQuery选择器所指向的DOM对象不同外，单选按钮和复选框的属性、选项、方法和属性都一样，所以在此不再赘述它们了。如果需要，读者可以参阅单选按钮的相关内容。

10.5　滑块

一般情况下，每次移动滑块，数字会增加1。如果滑块数值范围很大，比如从0到10000，而移动设备的屏幕宽度固定，则每次移动滑块时，数字增长可能比较快。jQuery Mobile的滑块样式如图10-6所示。

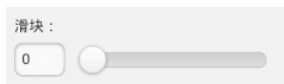


图10-6　滑块

滑块可以设置最小值和最大值，以约束滑块的数据范围。在滑块对象中，min属性用于设定最小值，max属性用于设定最大值，value属性用于设定默认值。

图10-6所示的滑块的代码如代码清单10-5所示。

代码清单10-5　滑块

```
<div data-role="fieldcontain">
  <label for="slider">滑块：</label>
  <input type="range" name="slider" id="slider" value="0" min="0" max="100"  />
</div>
```

注意　滑块相关的属性、选项、方法和事件与开关按钮是完全相同的，它们的唯一区别是用户界面呈现样式上。如果开发者希望了解开关按钮的属性、选项、方法和事件的用法，请参阅本节。

10.5.1 属性与选项

在尺寸和主题风格设定方面，滑块的属性与选项所实现的功能大致类似。表10-4列出了滑块属性、选项及其功能的对应关系。

表10-4 滑块属性与选项

选 项	属 性	功 能
mini	data-mini	设置标准尺寸或者mini尺寸，默认为false
theme	data-theme	设置主题风格。默认为空，主题风格继承自父级容器
trackTheme	data-track-theme	设置滑块轨道主题风格。默认为空，继承自父级容器

通过属性设置滑块轨道主题风格的代码如下：

```
<input type="range" id="mySlider" value="0" min="0" max="100" data-track-theme="e"/>
```

通过选项设置滑块轨道主题风格的代码如下：

```
$('#mySlider).slider({ trackTheme: "e" });
```

此外，滑块中有一些选项是没有对应属性的，如表10-5所示。

表10-5 滑块选项

选 项	功 能
disable	设置禁用滑块，默认值为false，表示不禁用
highlight	设置高亮显示，默认值为false，表示不高亮显示
initSelector	自定义渲染为滑块效果的选择器范围

如果需要通过属性禁用滑块，可以设置滑块的CSS样式为`ui-disabled`，而不是通过设置某个特定的禁用属性以实现，相关代码如下：

```
<input type="range" id="mySlider" value="0" min="0" max="100" class="ui-disable"/>
```

10.5.2 方法与事件

在滑块相关的操作中，主要涉及启用、禁用和刷新功能，相关方法及作用如表10-6所示。

表10-6 滑块方法

方 法	作 用
enable	启用滑块
disable	禁用滑块
refresh	刷新滑块样式

以禁用滑块为例，下面这段代码实现了对某个滑块的禁用操作：

```
$('#mySlider').slider('disable');
```

在jQuery Mobile 1.2.0之前，滑块只支持create事件，这个事件在滑块创建时触发。

从jQuery Mobile 1.2.0之后，滑块增加了两个新的事件——sliderstart和sliderstop，它们分别表示滑块调整开始和滑块调整结束。

代码清单10-6实现了sliderstart和sliderstop这两个事件。当开始拖动滑块时，将会触发sliderstart事件，并提示用户“开始移动滑块”。当滑块移动停止时，将会触发sliderstop事件，并提示用户“滑块移动结束”。执行效果如图10-7所示。

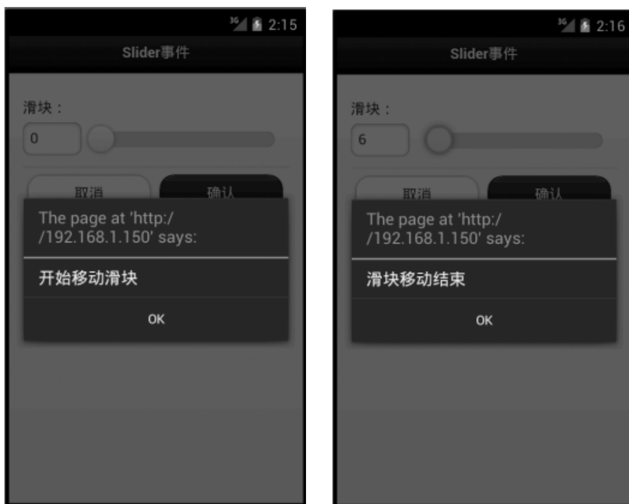


图10-7　滑块移动事件

代码清单10-6　实现sliderstart和sliderstop这两个事件的代码

```
$(document).ready(function(e) {
  $('#slider').bind('slidestart',function(event) {
    alert('开始移动滑块');
  });

  $('#slider').bind('slidestop',function(event) {
    alert('滑块移动结束');
  });
});
```

10.6　开关按钮

开关按钮的界面效果如图10-8所示，点击之后，即可切换开关值，其功能类似于单选按钮或者下拉菜单的功能。从用户体检的角度来说，开关按钮更加直观。

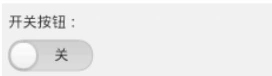


图10-8 开关按钮

实现开关按钮的代码如代码清单10-7所示。

代码清单10-7 开关按钮

```
<div data-role="fieldcontain">
  <label for="slider2">开关按钮：</label>
  <select name="slider2" id="slider2" data-role="slider">
    <option value="off">关</option>
    <option value="on">开</option>
  </select>
</div>
```

从开发者的角度来说，在jQuery Mobile中，开关按钮和滑块在属性、选项、方法和事件等方面的用法都是一样的，这里就不再赘述了。如果读者想了解相关信息，可以参考10.5节。

10.7 选择菜单

选择菜单从功能上更接近于单选按钮或者复选框，用户可以在一定范围内选择特定的内容，并随着表单而提交它们，如图10-9所示。与单选按钮或者复选框的呈现样式不同，选择菜单并不将所有内容直接呈现在浏览器上，而只有用户点击选择菜单时，备选内容才会呈现出来。

注意 选择菜单通常不会用作导航，如果需要实现导航菜单的功能，可以使用弹出页面中的弹出菜单来实现。如果想了解弹出菜单的相关内容，可参考第5章。



图10-9 选择菜单

图10-9所示的选择菜单的代码可参见代码清单10-8所示。

代码清单10-8　选择菜单

```
<div data-role="fieldcontain">
  <label for="select-choice-1" class="select">常用语言：</label>
  <select name="select-choice-1" id="select-choice-1">
    <option value="PHP">PHP</option>
    <option value="Python">Python</option>
    <option value="VB">Visual Basic</option>
    <option value="CSharp">C#</option>
  </select>
</div>
```

在实现选择菜单的时候，通常包含这样两部分内容：

- 作为选择菜单容器的select容器；
- 以option DOM对象表示的选择菜单的菜单项。

在选择菜单中，如果包含有对选择菜单进行说明的标签，则需要在标签中说明相关的选择菜单，例如`<label for="select-choice-1" class="select">`。

如果需要禁用原生菜单风格，可以将data-native-menu属性设置为false，呈现效果如图10-10所示。



图10-10　禁用data-native-menu属性的原生风格选择菜单

注意　菜单的原生风格和其他表单元素的原生风格设置是不同的，其中前者是一种jQuery Mobile的用户界面风格，后者则是HTML表单元素未经任何修饰的风格。选择菜单的原生风格是通过设定data-native-menu为false实现的，而其他表单元素的原生风格是通过设定data-role属性为none实现的。

图10-10所示的原生风格选择菜单的实现代码如代码清单10-9所示。

代码清单10-9　原生风格菜单

```
<div data-role="fieldcontain">
  <label for="select-choice-a" class="select">列表菜单：</label>
  <select name="select-choice-a" id="select-choice-a" data-native-menu="false">
    <option>选择你钟爱的语言</option>
    <option value="PHP">PHP</option>
    <option value="Python">Python</option>
    <option value="VB">Visual Basic</option>
    <option value="CSharp">C#</option>
  </select>
</div>
```

10.7.1　分组显示菜单项

选择菜单中的内容可以分组显示，示例图如图10-11所示。经过分组之后，一类内容被归纳在一起，这样有助于用户在不同分组中进行快速选择。每个分组菜单的标题和这个分组中的菜单项存在大致一个字符的缩进，这样使用者可以一目了然地识别出菜单的不同分组。如果分组之后的内容比较多，用户也可以通过上下滑动屏幕来看到更多菜单中的内容。

为了有助于使用者方便地识别出分组，不同分组的内容会缩进显示。例如在图10-11中，“页面开发”这组内容包含有HTML、JavaScript和CSS这3个菜单项，它们比“页面开发”这几个字向右缩进了几个像素。

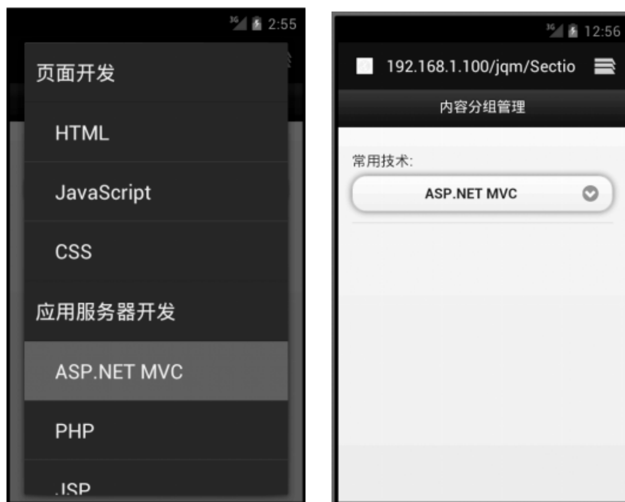


图10-11　左图为分组显示菜单内容，右图为选择某个菜单项后的效果

要实现菜单内容的分组，开发者需要将各个分组菜单项依次放置在optgroup容器中，然后再

将optgroup按顺序放在select容器中就可以了。optgourp的label属性值将会作为分组名称显示在菜单分组的列表中。

图10-11所示的分组的实现方式如代码清单10-10所示。

代码清单10-10　分组显示菜单内容

```
<div data-role="fieldcontain">
  <label for="select-choice-nc" class="select">常用技术：</label>
  <select name="select-choice-8" id="select-choice-nc">
    <optgroup label="页面开发">
      <option value="HTML">HTML</option>
      <option value="JavaScript">JavaScript</option>
      <option value="CSS">CSS</option>
    </optgroup>

    <optgroup label="应用服务器开发">
      <option value="ASP.NET MVC">ASP.NET MVC</option>
      <option value="PHP">PHP</option>
      <option value="JSP">JSP</option>
    </optgroup>

    <optgroup label="数据库">
      <option value="MySQL">MySQL</option>
      <option value="SQL Server">SQL Server</option>
      <option value="SQLite">SQLite</option>
    </optgroup>

    <optgroup label="操作系统">
      <option value="Linux">Linux</option>
      <option value="Windows">Windows</option>
      <option value="Android">Android</option>
    </optgroup>
  </select>
</div>
```

10.7.2　垂直分组与水平分组

如果选择菜单中的几项具有一定的相关性，那么分组则更加有助于使用者来进行选择和使用。通常，菜单文字较多时选择垂直分组，文字较少时选择水平分组，示例图如图10-12所示。不过即便是垂直分组，文字长度最好也不要过长。在垂直分组中，菜单的文字最好不要超过移动设备浏览器屏幕的宽度。

要想实现选择菜单的垂直分组，需要将相关选择菜单包含在fieldset容器中，并设置data-role属性为controlgroup。图10-12所示的垂直分组的实现代码如代码清单10-11所示。



图10-12　垂直分组与水平分组

代码清单10-11　垂直分组

```
<div data-role="fieldcontain">
  <fieldset data-role="controlgroup">
    <legend>垂直分组</legend>

    <label for="verticallygrouped-year">年</label>
    <select name="verticallygrouped-year" id="verticallygrouped-year">
      <option>年</option>
      <option value="2012">2012</option>
      <option value="2011">2011</option>
    </select>

    <label for="verticallygrouped-month">月</label>
    <select name="verticallygrouped-month" id="verticallygrouped-month">
      <option>月</option>
      <option value="01">01</option>
      <option value="02">02</option>
    </select>

    <label for="verticallygrouped-day">日</label>
    <select name="verticallygrouped-day" id="verticallygrouped-day">
      <option>日</option>
      <option value="01">01</option>
      <option value="02">02</option>
    </select>
  </fieldset >
</div>
```

默认情况下，菜单分组为垂直分组。要实现水平分组，只需要在之前垂直分组的基础上，在fieldset容器中设置属性data-type为horizontal即可：

```
<fieldset data-role="controlgroup" data-type="horizontal">
```

水平分组对于移动设备浏览器的显示宽度有一定要求，如果显示文字较多，而移动设备浏览器的宽度较窄，则可能会影响呈现效果。

10.7.3 禁用某个菜单项

在某种场景下，我们可能需要禁用某个菜单项，此时单独设置这个菜单项的disabled属性为disabled即可。

代码清单10-12实现了禁用某个菜单项的功能，实现效果如图10-13所示。

代码清单10-12 禁用某个菜单项

```
<div data-role="fieldcontain">
  <label for="select-choice-1" class="select">常用语言：</label>
  <select name="select-choice-1" id="select-choice-1">
    <option value="PHP">PHP</option>
    <option value="Python" disabled="disabled">Python</option>
    <option value="VB">Visual Basic</option>
    <option value="CSharp">C#</option>
  </select>
</div>
```

在代码清单10-12中，由于Python这个菜单项被设置为disabled，所以菜单项呈现为灰色和不可用的状态，如图10-13所示。



图10-13 Python菜单项被禁用

10.7.4 多选菜单

类似于复选框，选择菜单也支持多选的功能，示例图如图10-14所示。当打开多选菜单时，用户可以在菜单列表的右侧选择多个菜单项。选择完成后，这些选中的内容将会呈现在选择菜单上，而此时多选菜单的右侧将呈现一个气泡提示，告诉使用者选择的菜单项的数量。



图10-14 左侧为多选菜单，右侧为多选菜单的选择结果

要实现多选菜单，需要在DOM元素<select>中包含如下两个属性设置。

- 将multiple属性设置为multiple（即multiple="multiple"），用以标记选择菜单支持多选。
- 将data-native-menu属性设置为false。如果不进行这样的设置，则无法在移动设备中从选择菜单返回之前的表单界面。设置这个属性之后，将可以在选择表单的头部添加一个Delete图标按钮，点击这个按钮将可以实现选项确认。

图10-14所示的多选菜单的实现代码如代码清单10-13所示。

代码清单10-13 多选菜单

```
<div data-role="fieldcontain">
  <label for="select-choice-nc" class="select">常用技术：</label>
  <select name="select-choice-1" id="select-choice-nc"
    multiple="multiple" data-native-menu="false">
    <optgroup label="页面开发">
      <option value="HTML">HTML</option>
      <option value="JavaScript">JavaScript</option>
      <option value="CSS">CSS</option>
    </optgroup>

    <optgroup label="应用服务器开发">
      <option value="ASP.NET MVC">ASP.NET MVC</option>
      <option value="PHP">PHP</option>
      <option value="JSP">JSP</option>
    </optgroup>

    <optgroup label="数据库">
      <option value="MySQL">MySQL</option>
      <option value="SQL Server">SQL Server</option>
      <option value="SQLite">SQLite</option>
```

```
    </optgroup>

    <optgroup label="操作系统">
      <option value="Linux">Linux</option>
      <option value="Windows">Windows</option>
      <option value="Android">Android</option>
    </optgroup>
  </select>
</div>
```

10.7.5　属性与选项

开发者可以使用选择菜单的属性和选项设定选择菜单的外形、图标和主题风格等样式，具体如表10-7所示。

表10-7　选择菜单的属性与选项对照表

选　　项	属　　性	功　　能
corners	data-corners	设置选择菜单按钮是否为圆角矩形。默认为true，为圆角矩形。如果设置为false，则为直角矩形
icon	data-icon	设置选择菜单按钮的图标样式，默认为向下按钮arrow-d
iconpos	data-iconpos	设置选择菜单按钮的图标位置，默认为右侧right
iconshadow	data-iconshadow	设置按钮阴影，默认为显示阴影true。如果设置为false，则不显示阴影
inline	data-inline	设置是否以内联方式显示选择菜单按钮。默认为null，不以内联方式显示，也可以设置为false表示这个含义。设置为true，表示以内联方式显示
mini	data-mini	设置标准尺寸或者mini尺寸。默认为false，表示以标准尺寸显示
nativemenu	data-native-menu	设置是否以原生样式显示菜单。默认为true，以原生样式显示菜单。设置为false时，表示不以原生样式显示菜单，此时可能导致显示速度变慢。但是在多选菜单中，必须将此选项设置为false才可以正常工作
preventFocusZoom	data-prevent-focus-zoom	当焦点位于输入框中时，禁止执行缩放操作，以免缩放后输入困难。在iOS平台中，默认为true，表示禁止缩放
shadow	data-shadow	设置选择菜单按钮的阴影效果。默认为true，表示显示阴影效果。设置为false时，表示不显示阴影效果
theme	data-theme	设置主题风格。默认为空，主题风格继承自父级容器

下面这行代码表示选择菜单不使用原生菜单样式、直角矩形按钮，按钮中的图标为星型图标：

```
<select id="selectMenu" data-native-menu="false" data-corners="false" data-icon="start">
```

通过选项设置选择菜单按钮样式的代码如下：

```
$('# mySelectMenu).selectmenu({ icon: "start" });
```

此外，还有两个特别的菜单选项，具体如下所示。

- initSelector选项：用以设置自定义的选择器。当使用initSelector设定选择器后，jQuery

Mobile就会在执行初始化的时候，将这些选择器所对应的DOM对象初始化为选择菜单。一旦设定了选择器，jQuery Mobile默认的选择菜单标签就可能不再起作用了。

- overlayTheme选项：用于设定基于对话框的选择菜单的覆盖层的主题风格。

10.7.6 方法与事件

在选择菜单相关的操作中，主要涉及启用、禁用、刷新、打开和关闭等功能，相应的方法如表10-8所示。

表10-8 选择菜单相关的方法

方 法	功 能
enable	启用选择菜单
disable	禁用选择菜单
refresh	刷新菜单样式
open	打开选择菜单
close	关闭选择菜单

以禁用选择菜单为例，下面这段代码实现了对某个选择菜单的禁用操作：

```
$('#mySelectMenu').selectmenu('disable');
```

选择菜单支持create事件，该事件在创建选择菜单时触发。

10.8 禁用表单元素

如果要禁用某个表单元素，则可以通过设置CSS为ui-disabled来实现，示例图如图10-15所示。在图10-15中，文本输入框和查询输入框被标记为us-disabled，所以它们呈现为灰色，无法输入或使用。表单元素被禁用之后，基于其之上的输入、事件、方法等操作都将被一同禁用掉。该操作的代码原型如下：

```
<div data-role="fieldcontain" class="ui-disabled">
```

图10-15所示的禁用表单元素的实现代码如代码清单10-14所示。

代码清单10-14 禁用表单元素

```
<div data-role="fieldcontain">
  <label for="name">文本输入框：</label>
  <input type="text" name="name" id="name" value="" class="ui-disabled"/>
</div>

<div data-role="fieldcontain" class="ui-disabled">
  <label for="search">查询输入框：</label>
  <input type="search" name="search" id="search" value=""/>
</div>
```



图10-15 表单元素被禁用

仔细看图10-15中两个输入框的界面，可以发现它们略有差别：第一个输入框的文本不是灰色的，而第二个文本和输入框都是灰色的。这是由于第一个文本输入框的元素被禁用，而标签元素并没有被设置为ui-disabled样式，所以在第一个输入框中，只有文本输入框被禁用。

在第二个查询输入框的代码中，由于禁用样式ui-disabled被应用于fieldcontain容器上。这个容器包含有标签和查询输入框两个部分，所有包含在这个fieldcontain容器中的表单元素都是被禁用的状态。所以，在这个查询输入框的实现效果中，查询输入框的文本以及输入框都被禁用而呈现为灰色。

10.9 隐藏标签

在很多应用场景中，移动应用界面的尺寸比较有限，这时候就需要尽量有效地利用移动设备浏览器的界面空间。jQuery Mobile提供了一种隐藏标签的技巧，在这种实现中，标签不被独立显示，而是在输入框中显示。

要实现隐藏标签，大致需要如下两步。

(1) 在需要隐藏标签的<div data-role="filedcontain">容器中加入class="ui-hide-label"标记。

(2) 在输入框中添加 placeholder属性，并将标签的内容赋值给该属性。

注意 placeholder不是jQuery Mobile新增加的属性，而是HTML5在表单功能上支持的新特性。如果开发者感兴趣，可以参考HTML5手册。

图10-16是支持表单元素隐藏标签和没有隐藏标签的效果对比。



图10-16 表单元素隐藏标签

图10-16所示的页面的实现代码如代码清单10-15所示。

代码清单10-15 隐藏标签

```
<p>包含 ui-hide-label 定义</p>
<div data-role="fieldcontain" class="ui-hide-label">
  <label for="name">文本输入框：</label>
  <input type="text" name="name" id="name" value="" placeholder="文本输入框"/>
</div>

<div data-role="fieldcontain" class="ui-hide-label">
  <label for="search">查询输入框：</label>
  <input type="search" name="search" id="search" value="" placeholder="查询输入框"/>
</div>

<hr/>

<p>没有包含ui-hide-label定义</p>
<div data-role="fieldcontain">
  <label for="name">文本输入框：</label>
  <input type="text" name="name" id="name" value="" placeholder="文本输入框"/>
</div>

<div data-role="fieldcontain">
  <label for="search">查询输入框：</label>
  <input type="search" name="search" id="search" value="" placeholder="查询输入框"/>
</div>
```


10.10 mini 尺寸的表单样式

在一些移动应用开发场景中，例如折叠内容块、工具栏或列表视图中，如果使用正常尺寸布

置表单元素，会变得拥挤和局促，此时如果能将小一号的表单元素放入折叠内容块、工具栏或者列表视图中，用户界面的呈现效果将会更加舒适。jQuery Mobile提供了一套小尺寸的表单元素用于这种场景，它就是mini尺寸的表单样式。

注意　在jQuery Mobile 1.1.1及其之前的版本中，工具栏按钮默认为正常尺寸。自jQuery Mobile 1.2.0之后，工具栏按钮将默认为mini尺寸。要实现mini尺寸的表单元素，可以在表单元素中设置`data-mini`属性为`true`。

图10-17是一张对比图，显示了正常尺寸表单元素和mini尺寸表单元素的不同。图10-17右图所示的mini尺寸的登录表单的实现代码如代码清单10-16所示。



图10-17　左图为正常尺寸，右图有mini尺寸的登录输入框

代码清单10-16　mini尺寸表单元素

```
<div data-role="fieldcontain">
  <input type="text" id="name" value="" placeholder="登录名" data-mini="true"/>
  <input type="Password" id="password" value="" placeholder="密码" data-mini="true"/>
  <div style="margin-top:20px;" >
    <fieldset class="ui-grid-a">
      <div class="ui-block-a">
        <button type="reset" data-theme="d">取消</button>
      </div>
      <div class="ui-block-b">
        <button type="submit" data-theme="a">提交</button>
      </div>
    </fieldset>
  </div>
</div>
```

10.11 高级开发技术

Web移动应用和面向桌面浏览器的Web桌面应用都会经常用到表单技术，在本节中，我们将介绍两个表单提交的应用场景——表单验证和文件上传。在表单验证中，用户在提交内容时，将会首先进行内容验证。只有当内容验证通过后，才会将表单内容提交到服务器。在文件上传中，将会介绍如何在jQuery Mobile环境中实现移动设备文件向服务器的上传，例如上传一张照片或者视频到服务器。

10.11.1 表单验证

在基于jQuery开发的应用中，jQuery Validation是一种广泛用于表单验证的插件。开发者可以定制验证规则，jQuery Validation将会验证所提交的表单内容是否符合规则要求。如果符合规则要求，则可以提交到服务器，否则将会提示用户进行表单内容的修正。

在开发jQuery Mobile移动应用时，我们也可以通过jQuery Validation插件实现表单验证，如图10-18所示。在这个应用场景中，只有输入的登录内容符合一定规则（比如登录名和密码都不能为空），登录表单的内容才会被提交到服务器。



图10-18 表单验证

在jQuery Mobile应用开发中，要使用jQuery Validation插件实现表单内容验证，大致需要经过这样几个步骤。

(1) 在页面中除了引用jQuery和jQuery Mobile JavaScript库之外，还需要引用jQuery Validate插件。这里我们使用的验证插件代码下载自http://bassistance.de/jquery-plugins/jquery-plugin-validation/。

(2) 在需要进行验证的表单元素中通过class设置规则。例如，内容不得为空，则设置class为required。

(3) 在JavaScript中设置所要验证的表单内容并对错误提示进行样式定义。

代码清单10-17是表单验证的JavaScript实现与错误文字的样式定义。

代码清单10-17　表单验证

```
<link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
<script src="../js/jquery-1.7.1.min.js"></script>
<script src="../js/jquery.validate.js"></script>
<script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
<script type="text/javascript">
  $(document).ready(function(e) {
    $( "#formLogin" ).validate({
      submitHandler: function( form ) {}
    });
  });
</script>
<style type="text/css">
  label.error {
    color: red;
    font-size: 16px;
    font-weight: normal;
    line-height: 1.4;
    margin: 0.5em 0em 1em 0em;
    width: 100%;
    float: none;
    display: block;
  }
</style>
```

在这段代码中，需要被验证的表单DOM对象id为formLogin，验证插件也是绑定在这个表单对象上。在呈现错误消息的时候，label.error对错误消息进行了定义。

代码清单10-18是对输入内容进行验证的表单元素代码。

代码清单10-18　定义表单验证

```
<input type="text" name="name"class="required" placeholder="登录名" />
<input type="Password" name="password" class="required" placeholder="密码"/>
```

不同移动设备的分辨率相差很大，例如智能手机和高分辨率屏幕的平板电脑分辨率相差会很大。建议开发者使用媒体查询技术定义不同分辨率下的label.error显示方式。以登录错误为例，在手机中可以在独立一行中显示错误信息，而在平板电脑中则可能将输入框和错误提示信息放在同一行显示。

10.11.2　文件上传

jQuery Mobile应用可以实现基于表单的文件上传操作。但是与桌面浏览器实现文件上传有所不同。

由于jQuery Mobile默认通过Ajax方式实现页面加载，而文件上传则需要将内容提交到服务器上。所以需要在表单中设置data-ajax为false，禁用ajax方式进行表单提交。同桌面浏览器内容上传一样，也需要设置内容上传的编码方式enctype为multipart/form-data。

在进行内容选择的时候，受到操作系统安全性限制，并不是任何文件都可能会被选中并上传，只有特定的用户文件才可能被选中上传。

在图10-19中，点击左图的“Choose file”按钮选择文件，并点击提交按钮，可以将选择的文件提交到服务器上。图10-19的右图是可以选择上传文件的范围，包括照相机，录音，相册等。

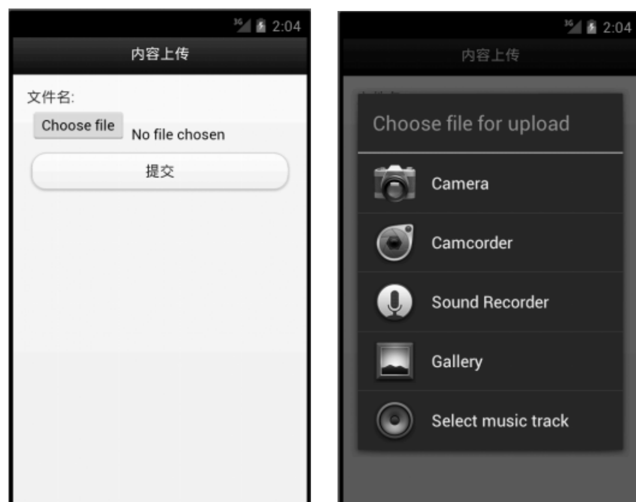


图10-19　左图为jQuery Mobile的内容上传界面，右图为浏览器内容浏览与选择界面

实现文件上传如代码清单10-19所示，所实现的文件上传界面效果，参见图10-19所示。

代码清单10-19　文件上传

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="列表视图演示">
      <header data-role="header">
        <h1>内容上传</h1>
      </header>

      <div class="content" data-role="content">
```

```
          <form action="upload_file.do" method="post" data-ajax="false"
            enctype="multipart/form-data">
            <label for="file">文件名:</label>
            <input type="file" name="file" id="file" />
            <br />
            <input type="submit" name="submit" value="提交" />
          </form>
        </div>
      </section>
    </body>
  </html>
```

第 11 章

页面布局与呈现

11

对于不同的移动设备，其分辨率不同，屏幕方向也不同，这些都需要移动应用提供不同的页面布局支持。缺少针对不同屏幕分辨率的设计，移动应用的页面元素在一些设备上会拥挤在一起，而在另一些设备中则可能会松松垮垮。jQuery Mobile基于HTML5提供了多种不同的页面布局与呈现技术来帮助开发者快速开发出针对不同移动设备的应用来。

在这一章中，我们将了解以下内容：

- 适应不同的分辨率；
- 改变屏幕方向；
- 分栏布局；
- 可折叠内容块；
- 折叠组。

11.1 适应不同的分辨率

不同的移动设备具有不同的屏幕尺寸和分辨率，这为移动应用开发带来一系列挑战。例如，当加载某个应用的背景图片或进行界面布局时，不同分辨率所适合的背景图片或者页面布局方式也会不同。将尺寸过小的图片应用到高分辨率的移动设备时，使用者会感到图片粗糙而不够精细；将尺寸过大的背景图片应用到低分辨率、小尺寸的屏幕时，使用者需要等待更长的时间才可以加载完这张图片。这一系列问题都是移动应用开发者所需要考虑的。

此外，一些移动设备支持水平方向（landscape）或者竖直方向（portrait）的显示，并且当屏幕发生旋转时浏览器也会进行相应的旋转。在不同的显示方向下，也需要移动应用能够支持不同的显示样式，例如文字列表或者表单的样式。面对不同的移动设备分辨率和使用场景，开发者可以通过采用CSS的媒体查询技术来定义移动应用的布局、图片和样式。

11.1.1 视口

视口（viewport）是计算机图形学中的一个重要概念，表示浏览器画布中可见的矩形区域。在jQuery Mobile中，视口就是移动设备浏览器在屏幕中呈现的部分。需要注意的是，在用jQuery Mobile开发移动应用时，必须设置视口，否则所有页面内容将会挤在一起而无法正常浏览。

注意　在HTML的meta标签中，viewport就是指浏览器所能显示的区域，在一些资料中称为可视窗口（visual viewport）。本章中使用的视口均是指可视窗口。

在一些网页开发资料中，视口的概念是整个网页的可视化区域，不但包括浏览器显示的部分，也包括超出浏览器的部分，这样的视口称为布局窗口（layout viewport）。

在图11-1中，我们标记出的矩形范围就是视口的作用范围。



图11-1　口的作用范围

作为HTML的meta元素，viewport用于设定浏览器屏幕的宽度以及初始的显示缩放比例。在本书的示例代码中，viewport通常被声明为：

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

其含义为，宽度为设备宽度，初始化缩放比例为100%。

在很多场景下，如果不希望用户对页面进行手工缩放，开发者可以声明user-scalable=no。

常见的viewport参数及其含义如表11-1所示。

表11-1　viewport参数及其含义

属　性	含　义
width	设置视口的宽度，其语法为width = device-width或者width = n。 前者表示视口的宽度为移动设备浏览器显示区域的宽度。 后者表示显示宽度为某个设定的像素值，该像素值的范围为200到10000，默认值为980像素
height	设置视口的高度，其语法为height = device-height或者height = n。 前者表示视口的高度为移动设备浏览器显示区域的高度。 后者表示显示高度为某个设定的像素值，该像素值的范围为223~10000

（续）

属　性	含　义
minimum-scale	设置视口被缩放的最小尺寸。 所设置的数值为浮点型数字，最小值为0.0，最大值为10.0，默认值为minimum-scale = 0.25。 设置视口被缩放的最大尺寸。 所设置的数值为浮点型数字，最小值为0.0，最大值为10.0，其默认值为maximum-scale = 1.6
initial-scale	设置初始视口的缩放比例。 所设置的数值为浮点型数字，介于minimum-scale与maximum-scale之间，例如initial-scale=1.0
user-scalable	设置是否允许对浏览器页面执行缩放操作。 默认情况下，user-scalable = yes表示允许执行缩放。如果设置为user-scalable = no，则表示禁止对页面执行缩放

如果没有设置viewport，界面将会进行缩放并呈现全部内容。以“Hello World!”页面为例，如果不设置viewport，那么页面将会被变成如图11-2所示的样子。显然，如果不设置viewport，页面内容将会被压缩得很小以至于无法正常阅读，这就是在移动页面中必须设置viewport的原因。



图11-2　没有设置viewport的“Hello World!”页面

11.1.2　媒体查询

移动设备浏览器和桌面浏览器的内容布局设计有很多不同，例如基于jQuery Mobile开发的应用一般需要同时支持水平和垂直方向这两种显示方式。此外，移动应用还需要支持各种分辨率下的移动设备，从3.7寸的手机浏览器到12寸的平板电脑。

对于移动应用而言，当移动设备的方向或者分辨率不同时，移动应用可以相应加载不同的CSS定义，并根据不同的CSS定义呈现不同的布局、字体、字号和配色等。在jQuery Mobile应用中使用媒体查询（media query）技术的初衷就是为了支持前面所提到的这些场景。

媒体查询技术可以基于当前移动设备浏览器窗口尺寸的不同而动态加载不同的CSS文件或CSS定义。如果程序运行在桌面浏览器，那么媒体查询也可以基于桌面浏览器窗口尺寸的不同而加载不同的CSS。例如，在低分辨率的移动设备下，基于媒体查询技术所加载的CSS布局是单栏布局，而在高分辨率的移动设备下则自动变成多栏布局。此外，在高分辨率环境下的背景图片也可能会加载更高分辨率的图片。

在媒体查询中，为不同的分辨率加载不同CSS的边界值也叫做折断点。在大多数应用场景下，媒体查询的折断点需要使用相对尺寸em进行设置，而不可以使用像素。在移动设备中，尽管浏览器的尺寸可能是相同的，但是因为移动设备屏幕的分辨率不同，浏览器的水平和竖直方向的像素数量差异会很大。如果以像素来设置折断点，则在高分辨率屏幕下界面呈现可能会出现异常。

注意　自jQuery Mobile 1.3.0版本以来，它开始支持响应式设计，这也是建立在媒体查询的基础之上的。基于响应式设计，移动应用程序将自动根据移动设备的分辨率而组织表格的排版和内容呈现方式，具体内容请参见第12章。

下面这段代码通过媒体查询技术识别屏幕不同的显示方向，从而加载不同的CSS文件：

```
<!-- 竖直方向显示，则加载portrait.css文件 -->
<link rel="stylesheet" media="all and (orientation:portrait)" href="portrait.css" />
<!-- 水平方向显示，则加载landscape.css 文件 -->
<link rel="stylesheet" media="all and (orientation:landscape)" href="landscape.css" />
```

其中portrait.css文件的代码如下：

```
body{
  background-color:#FFF;
  color:#000;
}
```

landscape.css文件的代码如下：

```
body{
  background-color:#000;
  color:#FFF;
}
```

当屏幕方向为竖直方向时，portrait.css文件的内容被加载，此时为黑色背景、白色文字。当屏幕转向水平方向时，landscape.css文件的内容被加载，屏幕显示变为白色背景、黑色文字。这样的显示效果变化是不需要重新下载CSS文件的，而是在触发媒体查询的时候由浏览器自动加载不同CSS文件的内容并实现页面呈现的。

小经验　在使用Android模拟器开发移动应用时，可以使用快捷键触发屏幕旋转，比如按Ctrl+F11组合键，模拟器按照顺时针旋转，按Ctrl+F12组合键，模拟器按照逆时针旋转。

使用媒体查询方法加载不同CSS文件的示例代码参见代码清单11-1。

代码清单11-1　基于媒体查询技术加载不同的CSS

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <!-- 竖直方向显示，则加载portrait.css文件 -->
    <link href="MediaQueries01/portrait.css" rel="stylesheet" type="text/css"
      media="all and (orientation:portrait)" />
    <!-- 水平方向显示，则加载landscape.css文件 -->
    <link href="MediaQueries01/landscape.css" rel="stylesheet" type="text/css"
      media="all and (orientation:landscape)" />
  </head>
  <body>
    不同屏幕方向下，基于媒体查询技术所呈现的文字颜色和背景色是不同的。
  </body>
</html>
```

运行上述代码，得到的运行结果如图11-3和图11-4所示。



图11-3　竖直方向下媒体查询呈现效果



图11-4　水平方向下媒体查询呈现效果

对于不同的显示设备，例如不同分辨率的手机或平板电脑，可以通过识别不同的屏幕宽度来加载不同的CSS文件，示例代码如下：

```
<!-- 最小屏幕宽度为481px时，则加载style.css文件 -->
<link rel="stylesheet" href="style.css" media="screen and (min-device-width: 481px)" type="text/css" />
<!-- 最大屏幕宽度为480px时，则加载style_mobile.css文件 -->
<link type="text/css" rel="stylesheet" media="only screen and (max-device-width: 480px)"
href="style_mobile.css" />
```

需要注意的是，如果所开发的应用面向移动设备、平板电脑和桌面浏览器等多种运行环境，媒体查询的折断点写法会略有区别。通常，`only screen and (max-device-width: 480px)`多用于移动设备，而`screen and (min-device-width: 481px)`多用于常见的桌面浏览器或平板电脑。这是一种基于经验的写法，开发者会针对移动设备浏览器和桌面浏览器开发不同的CSS布局和配色等。当浏览器宽度小于480像素时，则使用为移动设备所开发的CSS布局和配色。而如果浏览器的宽度超过480像素，则使用为桌面浏览器和平板电脑所开发的CSS布局和配色。也正是因为“480像素”的设定是基于经验的，开发者可以根据实际应用场景选择合适的CSS媒体查询规则。例如，如果需要同时支持高分辨率屏幕的移动设备，则可能需要使用相对尺寸em，而不能用像素来设置折断点。

对于早期版本的IE浏览器，可能不支持媒体查询技术。作为一种变通的方法，可以写成这样的形式。

```
<!--[if IE]><link rel="stylesheet" type="text/css" href="style.css"  media="screen" /><![endif]-->
```

事实上，在面向移动设备的开发中，`<!--[if IE]>…<![endif]-->`的写法通常很少用到，这主要是因为在移动设备中很少有人使用IE。

小经验　早期的jQuery Mobile测试版本通过Media Query Helper类来识别屏幕方向，而在jQuery Mobile 1.0之后，我们通过CSS3的媒体查询技术提供支持，而不再使用Media Query Helper类。如果开发者是从早期的jQuery Mobile升级或者迁移到新版本，需要留心这个变化。阅读jQuery Mobile技术资料的时候，建议留心范例代码所使用的jQuery Mobile版本。

11.1.3　背景图片进阶

基于不同的屏幕分辨率或者屏幕方向，通过媒体查询技术选择加载不同的CSS文件，是CSS3之后的新特性，这个特性还经常用于加载合适尺寸和比例的背景图片。例如，可以在分辨率更高的屏幕下加载更高分辨率的图片。

要处理CSS3的背景图片，通常还会用到这几个属性：background-origin、background-clip和background-size。

- background-origin：用于定位背景图片的位置，其语法为padding-box | border-box |content-box，其中各个属性及其含义如表11-2所示。

表11-2　background-origin属性

属　　性	含　　义
padding-box	背景图片相对于Padding Box中的定位
border-box	背景图片相对于Border Box中的定位
content-box	背景图片相对于Content Box中的定位

- background-clip：用于指定背景颜色的区域，其语法为background-clip: border-box |padding-box | content-box，其中各个属性及其含义如表11-3所示。

表11-3　background-clip属性

属　　性	含　　义
padding-box	背景颜色从Padding Box处裁剪
border-box	背景颜色从Border Box处裁剪
content-box	背景颜色从Content Box处裁剪

- background-size：用于设定背景图片的尺寸，其语法为length | percentage | cover | contain，其中各个属性及其含义如表11-4所示。

表11-4　background-size属性及其含义

属　　性	含　　义
length	设定背景图片的高度与宽度，第一个值是宽度，第二个是高度。如果只给出一个值，则第二个值默认设置为auto。例如background-size: 200px 200px和background-size: 200px
percentage	基于父级元素的尺寸，以百分比设置背景图片的宽度和高度。第一个值是宽度，第二个是高度。如果只给出第一个值，则第二个值默认设置为auto。例如 background-size: 70% 70%和background-size: 80%
cover	缩放图片，以完全填充整个背景区域。在这个设定之下，部分图片可能会超出背景区域的范围
contain	缩放背景图片，以固定长宽比例填充背景区域。在这个设定下，如果背景区域的比例和图片比例不一致，则可能会出现留白

11.2　改变屏幕方向

移动设备的宽度和高度发生转换，可能是由于移动设备的方向发生变化而产生的。如果移动设备的屏幕方向发生改变，将会触发orientationchange事件，例如设备屏幕从水平方向转为竖直方向时，将执行如下代码：

```
$(window).bind('orientationchange', function(){
  alert('屏幕方向发生旋转');
});
```

如果要禁用orientationchange事件，可以在jQuery Mobile页面中通过设置$.mobile.orientationChangeEnabled = false来实现。

小经验　在Android浏览器中，当触发orientationchange事件时，screen.width和screen.height属性的值会同时发生变化，但是在一些手机（例如iPhone）中却可能不会发生。

代码清单11-2实现了响应屏幕方向改变事件的代码。在这个事件中，屏幕的宽度和高度会通过消息框呈现出来。而在不同的屏幕方向下，Android移动设备的宽度和高度会发生互换。

代码清单11-2　屏幕方向旋转

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script type="text/javascript">

      //取消下面这行注释，可以观察orientationChangeEnabled设置的结果
      //$.mobile.orientationChangeEnabled = false;

      $(window).bind('orientationchange', function(){
        alert('屏幕方向发生改变。\n屏幕宽度 '+screen.width+'像素\n屏幕高度 '
          +screen.height+'像素');
      });
    </script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
        <h1>jQuery Mobile跨平台移动应用</h1>
      </header>
      <div class="content" data-role="content">
```

```
        <p>当屏幕方向发生改变时，将触发orientationchange事件。</p>
      </div>
      <footer data-role="footer"></footer>
    </section>
  </body>
</html>
```

运行代码清单11-2，得到的结果如图11-5所示。



图11-5　改变屏幕方向时，会触发orientationchange事件

11.3　分栏布局

在之前各章中，主要的排版布局是一栏自上而下将内容顺序列出。在这一节中，我们主要介绍一下分栏布局。使用分栏方式排版，将有助于在一个有限的屏幕空间内更有序地展示更多内容。

在移动设备浏览器显示尺寸比较小而显示内容为大段文字或图文混排为主的页面中，通常不会使用多栏排版。在高分辨率的移动设备浏览器中，分栏显示有助于更好地利用屏幕空间，提升用户体验。因此，有时候，使用的排版布局与目标受众的设备种类相关。

注意　在一些场景下，移动设备的分辨率跨度很大，这就需要用到用户界面的响应式设计来帮助改善移动应用界面设计，这将可能需要在不同分辨率的移动设备中使用不同的分栏布局。

对于传统的桌面浏览器，通常有两种方法来实现布局设计。

- 通过CSS与div集成的方式实现版式布局。
- 通过定制没有框线的<table>表格实现布局设计。

不论使用哪种方式来实现布局，都可以设计富有表现力的布局。然而，这些并不完全适合移动应用的使用场景。虽然当下主流的移动设备浏览器可以显示大多数页面，不管这样的页面是为移动应用还是为桌面浏览器设计的。不过，移动设备的屏幕尺寸和计算能力毕竟有限，在移动设备上使用面向桌面浏览器所开发的应用并不方便。面向移动设备浏览器而优化移动应用，以更加

简洁、高效和实用的方式进行移动应用布局设计，这正是jQuery Mobile所实现的。

jQuery Mobile通过支持分栏布局，提供了简单而有效的界面排版方式。开发者可以使用jQuery Mobile原生的布局方式快速生成界面风格统一的分栏布局。当然，如果开发者需要更丰富的分栏方式，也可以使用其他方式进行定制化开发。

jQuery Mobile分栏布局是通过CSS定义实现的，主要包含两个部分，即栏目数量以及内容所在栏目的次序，具体如表11-5所示。

表11-5 定义分栏

功 能	语 法
定义栏目数量	ui-guid-a/b/c/d 例如<div class="ui-grid-a">…</div>表示二栏布局
定义内容块在栏目中的位置	ui-block-a/b/c/d/e 例如<div class="ui-block-b">…</div>表示内容被填充于第二栏

在表11-5中，ui-grid-a/b/c/d分别表示对应的<div>或<section>中的栏目数量，分别为二栏、三栏、四栏、五栏。而ui-block-a/b/c/d/e分别表示相应内容块位于第一栏、第二栏、第三栏、第四栏或者第五栏。

注意 这里的栏目数量是从两栏开始的，栏目数量的最大值是5栏，所以表示布局分为五栏的序号为d，CSS定义为ui-grid-d。

标记内容所在栏目的位置是从第一栏开始的，所以，第5栏所对应的为e，CSS会表示为ui-block-e。

这是开发者很容易疏忽的地方。

下面举例说明如何使用CSS定义实现两栏布局：

```
<div class="ui-grid-a">
  <div class="ui-block-a"><p>第一栏</p></div>
  <div class="ui-block-b"><p>第二栏</p></div>
</div>
```

运行上述代码，得到的运行结果如图11-6所示。

注意 在实际的应用中，分栏布局不会存在图11-6中的边框线。在上述示例中，为了能够明晰两栏布局的边界，我们在CSS中将内容的背景设置为白色，而将边框设置为一个像素宽度的黑色实线框。

在分栏布局中，各个内容的宽度通常是平均分配的。对于不同的栏数，各个分栏的宽度比例如表11-6所示。
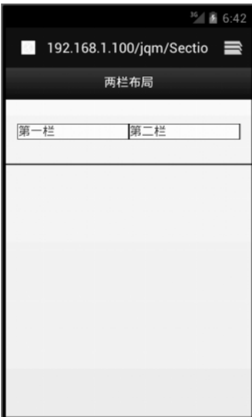


图11-6 两栏布局

表11-6 不同分栏所占的比例

分栏数量	每栏内容所占的宽度
二栏布局	50%
三栏布局	大约33%
四栏布局	25%
五栏布局	20%

分栏越多，每栏在屏幕中的尺寸就越小，这在移动应用开发中需要格外小心。如果在屏幕尺寸较小的手机浏览器上显示四栏或者五栏的布局，并且每个分栏中都是相对字数较多的文字或图片内容，则可能会因为界面呈现局促而降低用户体验。如果在多栏布局中，每个分栏包含的是一个含义清晰美观的图标按钮，则可能会赢得更好的用户体验。代码清单11-3完整地实现了两栏布局。

代码清单11-3 两栏布局的示例代码

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <style>
      /*自定义CSS，用以标识两栏布局边框范围*/
      .content div div p {
        background-color: #fff;
        border: 1px solid #000;
      }
    </style>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
```

```
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
        <h1>两栏布局</h1>
      </header>
      <div class="content" data-role="content">
        <div class="ui-grid-a">
          <div class="ui-block-a"><p>第一栏</p></div>
          <div class="ui-block-b"><p>第二栏</p></div>
        </div>
      </div>
      <footer data-role="footer"></footer>
    </section>
  </body>
</html>
```

如果需要移动应用支持更多的分栏，可以通过增加分栏来实现。在下面的五栏布局中，我们依次加入标记为ui-block-c到ui-block-e的CSS定义，实现了第三栏到第五栏的定义：

```
<div class="ui-grid-d">
  <div class="ui-block-a"><p>第一栏</p></div>
  <div class="ui-block-b"><p>第二栏</p></div>
  <div class="ui-block-c"><p>第三栏</p></div>
  <div class="ui-block-d"><p>第四栏</p></div>
  <div class="ui-block-e"><p>第五栏</p></div>
</div>
```

注意　在jQuery Mobile中使用这样的方式定义分栏数量，最多可以定义5个分栏。如果开发者需要更多的分栏布局，则需要自己开发CSS布局来实现。

如果分栏设计的内部还希望有多行的设计，例如两个4栏布局自上而下排列，通常并不需要重复设置多个<div class="ui-grid-b">…</div>标签，而只需顺序排列包含有ui-block-a/b/c/d/e定义的div即可。

图11-7所示的样式就是通过顺序排列ui-block-a/b/c中的各块内容实现的，其实现代码如代码清单11-4所示。

代码清单11-4　多行分栏布局

```
<div class="ui-grid-b">
  <div class="ui-block-a"><p>第一行<br/>第一栏</p></div>
  <div class="ui-block-b"><p>第二栏</p></div>
  <div class="ui-block-c"><p>第三栏</p></div>

  <div class="ui-block-a"><p>第二行<br />第一栏</p></div>
  <div class="ui-block-b"><p>第二栏</p></div>
  <div class="ui-block-c"><p>第三栏</p></div>

  <div class="ui-block-a"><p>第三行<br />第一栏</p></div>
  <div class="ui-block-b"><p>第二栏</p></div>
  <div class="ui-block-c"><p>第三栏</p></div>
</div>
```



图11-7　三栏多行布局

11.4　可折叠内容块

可折叠内容块是指特定标记内的图文内容或表单可以被折叠起来，它通常由两部分组成——头部按钮和可折叠内容。当用户需要操作的时候，直接点击头部按钮即可展开或者折叠所包含的内容，如图11-8所示。



图11-8　左侧为折叠状态，右侧为打开状态

由于移动设备的屏幕相对较小，字号通常也比较小，所以如果将内容全部展开，篇幅可能会很长，此时定位内容也需要手工不断翻屏，这将影响阅读体验。使用可折叠内容块，可以帮助用

户很快定位到相关主题，展开可折叠内容块之后就可以直接阅读或执行相应操作。因此，使用可折叠内容块改善了阅读体验。

使用jQuery Mobile建立的可折叠内容块通常由如下3部分组成。

- 定义data-role属性为collapsible的DIV DOM对象，用以标记折叠内容块的范围。
- 以标题标签定义可折叠内容块的标题。在可折叠内容块中，这个标题将呈现为一个用以控制展开或折叠的按钮。
- 可折叠内容块的内容。

实现图11-8所示的效果的代码参见代码清单11-5。

代码清单11-5　折叠代码

```
<div data-role="collapsible">
  <h3>折叠内容的头部按钮</h3>
  默认情况下，折叠内容被收起。当点击折叠内容的头部按钮时，则内容被打开。
</div>
```

这里我们使用了h3标题。事实上，任何h1到h6级别的标题在第一行都将呈现为折叠内容块的头部按钮。

通常，jQuery Mobile界面呈现不会因为采用了低级别的标题（例如h6）而导致可折叠内容块中头部按钮的字体或字号发生改变，例如：

```
<div data-role="collapsible">
  <h1>标记为h1的头部按钮</h1>
</div>
<div data-role="collapsible">
  <h6>标记为h6的头部按钮</h6>
</div>
```

这段代码的运行结果如图11-9所示，从中可以看到，h1与h6标题的呈现效果是一致的。



图11-9　可折叠内容块中h1和h6标题的呈现效果一致

11.4.1　嵌套可折叠内容块

虽然每个可折叠内容块只能作用于一个内容块区域，但是它也可以通过级联方式包含其他可折叠内容块，这是一种树状信息组织。不过，由于通过树状方式组织内容要求移动设备浏览器足够宽，否则无法正常展现一级级的树状结构，所有树状结构在移动设备的界面呈现中并不方便。相比之下，使用嵌套的可折叠内容块既可以有效地以类似的方式组织内容的结构，也能在有限的显示空间中获得不错的用户体验。嵌套可折叠内容块的示例图如图11-10所示。



图11-10　具有嵌套内容的可折叠内容块的折叠与展开效果图

图11-10的实现代码如代码清单11-6所示。

代码清单11-6　嵌套可折叠内容块

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
        <h1>可折叠内容块</h1>
      </header>
      <div class="content" data-role="content">
        <div data-role="collapsible" id="collapsible01" data-theme="e" data-content-theme='b'>
```

```
            <h3>折叠内容的头部按钮</h3>
            这里是最外层可折叠内容块的内容区域。
            <div data-role="collapsible" id="sub-collapsible" data-content-theme='c'>
              <h4>被嵌套的内容</h4>
              这里是被嵌套的内容
            </div>
          </div>
        </div>
        <footer data-role="footer"></footer>
      </section>
    </body>
  </html>
```

在实现具有嵌套关系的可折叠内容块时，有几个方面值得注意，具体如下所示。

- 外层嵌套可折叠内容块和内部可折叠内容块最好使用不同的主题风格，以便使用者分辨不同的可折叠内容块级别。
- 各层可折叠内容块通过声明data-content-theme属性定义内容区域的显示风格，这样的设置能在可折叠内容块的内容边界处出现一个边框线。这个边框线相对明显地分割了各级嵌套内容，方便使用者阅读内容块区域的内容。

11.4.2　属性

可折叠内容块可以通过定义DOM容器属性实现常用的设置而不需要开发负责的JavaScript程序，例如界面样式、内容块样式和标题文字等，如表11-7所示。

表11-7　可折叠内容块的属性

属　　性	功　　能

data-collapsed	设置为折叠状态或展开状态，其默认值为true，表示折叠状态；如果其值为false，则为展开状态。
	示例代码：
	`<div data-role="collapsible"data-collapsed="true">`
	效果为：
	示例代码：
	`<div data-role="collapsible"data-collapsed="false">`
	效果为：

data-mini	jQuery Mobile 1.1.0开始支持这个属性，用于设置内容区域表单组件呈现为标准尺寸或者压缩尺寸。其默认值为false，表示以标准尺寸呈现。 如果将其值设置为true，则表单元素呈现为压缩尺寸。 示例代码： `<div data-role="collapsible"data-mini="true">` 效果为：

（续）

属　　性	功　　能
data-iconpos	设置可折叠内容块标题的图标位置，其可选值如下。 ● left：图标位于左侧，这是默认值。 ● right：图标位于右侧。 ● top：图标位于上方。 ● bottom：图标位于下方。 ● notext：理论上说，此种情况下文字会被隐藏而只显示图标。 各种效果的示例代码如下： <div data-role="collapsible" data-iconpos="left"> 效果为： 图标被设置在左侧 示例代码： <div data-role="collapsible" data-iconpos="right"> 效果为： 图标被设置在右侧 示例代码： <div data-role="collapsible" data-iconpos="top"> 效果为： 图标被设置在上方 示例代码： <div data-role="collapsible" data-iconpos="bottom"> 效果为： 图标被设置在下方
data-theme	设置可折叠内容块的主题风格，数值为a至z
data-content-theme	设置可折叠内容块内部区域的主题风格，数值为a至z

11.4.3　选项

通过可折叠内容块的选项设置,开发者可以在初始化过程中通过JavaScript程序对可折叠内容块进行样式定制，具体可参见表11-8。

表11-8　可折叠内容块的选项

选　　项	功　　能
collapsed	设置默认状态为折叠状态或展开状态，其默认值为true，表示可折叠内容块的默认状态为折叠状态；如果将其值设置为false，则为展开状态
mini	设置内容区域表单组件呈现为标准尺寸或者压缩尺寸。从jQuery Mobile 1.1.0开始支持这个选项，其默认值为false，表示以标准尺寸呈现。如果将其值设置为true，则表单元素呈现为压缩尺寸

（续）

选　项	功　能
iconpos	设置可折叠内容块标题图标的位置，主要有如下可选项。 ● left：图标位于文字左侧，这是默认值。 ● right：图标位于文字右侧。 ● top：图标位于文字上方。 ● bottom：图标位于文字下方。 ● notext：无文字而只有图标。
theme	设置可折叠内容块的主题风格，数值为a至z
contentTheme	设置可折叠内容块内部区域的主题风格，数值为a至z。 注意这个选项值和属性data-content-theme命名风格略有差别
collapseCueText	折叠操作的提示文字，其默认值为click to collapse contents（点击以折叠内容块）
expandCueText	展开操作的提示文字，其默认值为expand with a click（点击以展开内容块）
heading	设置显示的标题定义，其默认值为h1、h2、h3、h4、h5、h6、legend。 如果该选项被设置，而在可折叠内容块中没有标记呈现的标题，则可折叠内容块不会呈现
initSelector	设置选择器，以选择可以被可折叠内容块渲染的DOM容器

如果jQuery Mobile程序在初始化的时候指定了initSelector选择器所调取的属性，而在data-role="collapsible"的DOM容器却没有声明相应CSS属性的定义，则这个可折叠内容块将不会被渲染。在代码清单11-7中，由于initSelector选项被设置为.mycollapsible，所以只有设置这个CSS的class属性值的DOM容器，才可以被渲染成可折叠内容块。所以第一个可折叠内容块的DOM容器中的内容在界面中被呈现为可折叠内容块的样子。而第二个可折叠内容块因为没有设置class为.mycollapsible，所以没有被渲染成可折叠内容块的样子。

代码清单11-7　可折叠内容块initSelector选项示例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script type="text/javascript">
      $(document).bind('mobileinit', function(){
        $.mobile.collapsible.prototype.options.initSelector = ".mycollapsible";
      });
    </script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="page1" data-role="page">
```

```
        <header data-role="header">
          <h1>可折叠内容块</h1>
        </header>
        <div class="content" data-role="content">
          <div data-role="collapsible" data-collapsed="true" class="mycollapsible">
            <h3>Class被设置为mycollapsible</h3>
          </div>
          <div data-role="collapsible" data-collapsed="true">
            <h3>未被设置为mycollapsible </h3>
          </div>
        </div>
        <footer data-role="footer"></footer>
      </section>
    </body>
  </html>
```

运行上述代码，得到的运行结果如图11-11所示。可以发现，第二段内容没有被可折叠内容块渲染。



图11-11 应用initSelector选项的效果

使用可折叠内容块的heading选项时，需要开发者比较谨慎，因为只有被声明为可折叠内容块标题的内容才会成为标题，而其他文字则会按照系统默认定义的呈现方式进行呈现。

图11-12显示设置两种不同标题时的呈现效果。在这个设置中，可折叠内容块中有两个标题，按照通常情况，第一个标题<h1>将会被作为标题显示，但是因为heading选项所对应的CSS属性被设置在第二个标题上，所以第二个标题的内容被作为可折叠内容块的标题被呈现出来。图11-12的实现代码如代码清单11-8所示，读者可以留意其中<h1>和<h3>的顺序。



图11-12　应用heading选项的效果

代码清单11-8　应用heading选项不当的情况

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script type="text/javascript">
      $(document).bind('mobileinit', function(){
        $.mobile.collapsible.prototype.options.heading = ".mycollapsibleheading";
      });
    </script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="page1" data-role="page">
      <header data-role="header">
        <h1>可折叠内容块</h1>
      </header>
      <div class="content" data-role="content">
        <div data-role="collapsible" data-collapsed="true" data-content-theme='b'>
          <h1>不被显示的标题</h1>
          <h3 class="mycollapsibleheading">此标题被用于显示</h3>
          正文内容
        </div>
        <div data-role="collapsible" data-collapsed="true">
```

```
      <h3>由于未设置CSS属性，这段折叠内容块不会被显示</h3>
     </div>
     未包含在折叠内容块中的文字。
    </div>
    <footer data-role="footer"></footer>
   </section>
  </body>
</html>
```

通常，在可折叠内容块中没有声明heading选项的情况下，应该是第一个<h1>标签所包含的内容被呈现为可折叠内容块的标题，而不应该是第二个。但是对前面代码进行初始化的时候，声明特定class属性的内容才可以用作标题，所以第一个<h1>标签的内容没有呈现为标题，而在它之后的<h3 class="mycollapsibleheading">所包含的内容成了这个可折叠内容块的标题。

注意　这个应用场景其实包含两个折叠内容块，但是因为第二个折叠内容块没有声明用于显示的class，所以在最终的浏览器呈现界面中，第二个折叠内容块的内容没有显示出来。这是开发时需要小心的。

11.4.4　事件

可折叠内容块的事件用以响应操作行为，常用的事件主要包括3种，如表11-9所示。

表11-9　可折叠内容块的事件

事　　件	功　　能
create	可折叠内容块被创建时触发
collapse	可折叠内容块被折叠时触发
expand	可折叠内容块被展开时触发

只要打开包含有可折叠内容块的页面时，折叠或展开事件就会被触发一次。这个事件触发发生在可折叠内容块生成的时候，事件的触发也与是否手工展开或者折叠没有直接关系。所以，在进行可折叠内容块的事件绑定时，需要注意绑定程序的位置。

代码清单11-9是绑定可折叠内容块事件响应的代码片段，运行效果参见图11-13。

代码清单11-9　折叠内容块事件

```
$(document).ready(function(e) {
    $(document).delegate("#collapsible01","expand", function(){
        alert('内容被展开');
    });
    $(document).delegate("#collapsible01","collapse", function(){
        alert('内容被折叠');
    });
});
```

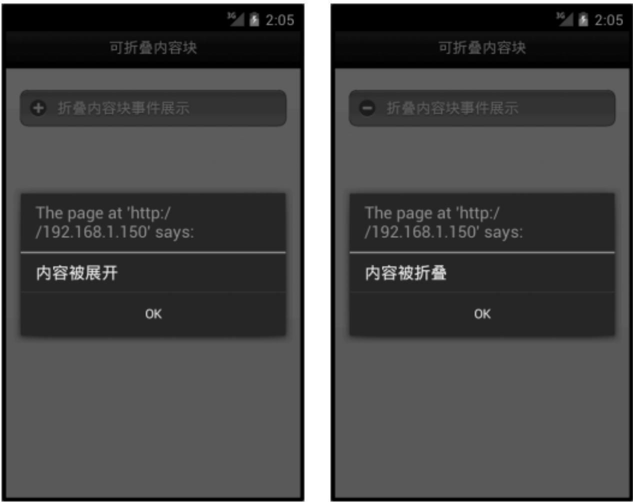


图11-13　左图为expand事件触发的效果，右图为collapse事件触发的效果

11.5　折叠组

折叠组（collapsible set）按照从上到下的顺序排列可折叠内容块，这使界面的整体感更强，示例图如图11-14所示。



图11-14　折叠组折叠与展开样式

图11-14的实现代码如代码清单11-10所示，在这个代码片段中，折叠组DOM容器中依次放置了3个可折叠内容块。

代码清单11-10 折叠组

```
<div data-role="collapsible-set">
  <div id="collapsible01" data-role="collapsible" data-content-theme='b'>
    <h3>可折叠内容块01</h3>
    可折叠内容块内容
  </div>

  <div id="collapsible02" data-role="collapsible" data-content-theme='b'>
    <h3>可折叠内容块02</h3>
    可折叠内容块内容
  </div>

  <div id="collapsible03" data-role="collapsible" data-content-theme='b'>
    <h3>可折叠内容块03</h3>
    可折叠内容块内容
  </div>
</div>
```

正如其名称所展现的那样，折叠组就是由若干个独立的可折叠内容块集合而成的。通过其代码结构，也可以看到这样的特点，若干可折叠内容块被一个标记为collapsible-set的容器集合在一起，并作为一个整体在移动设备浏览器中呈现。

由于折叠组是由若干独立的可折叠内容块按照顺序集和而成的，所以呈现出的属性、选项和事件与可折叠内容块非常类似。

值得注意的是，折叠组的事件和可折叠内容块的事件略有不同，它只有create事件而没有collapse和expand事件，这是由于折叠组是各个可折叠内容块的集合。对于这个集合而言，是不存在折叠和展开操作的，这样的操作是其内部可折叠内容块的操作。

折叠组通过refresh()方法刷新其内容，例如增加一个新的折叠内容块。

第 12 章

响应式设计

12

响应式设计是jQuery Mobile 1.3.0开始支持的新特性，用于在不同分辨率的移动设备上将最有价值的数据以良好的用户体验呈现出来。当屏幕在横屏和竖屏中切换时，或者当程序运行在不同分辨率的智能手机或者平板电脑时，响应式设计将会根据视口尺寸的不同通过HTML5的媒体查询技术加载不同的CSS定义而呈现出不同的用户界面。

在这一章中，我们将会了解到：

- 基于jQuery Mobile实现响应式设计；
- 分栏技术；
- 回流表格（Reflow Table）；
- 字段切换表格（Column-Toggle Table）；
- 滑动面板；
- 支持触控操作的滑动面板。

12.1 基于 jQuery Mobile 实现响应式设计

使用jQuery Mobile实现响应式设计时，通常有两种方法，具体如下所示。

- 基于媒体查询技术设置折断点，在不同的折断点（breakpoint）加载不同的CSS并呈现不同的用户界面效果，这也是使用最多的方法。
- 基于jQuery Mobile默认设置的回流表格、字段切换表格和滑动面板实现响应式设计。

如果使用第二种方法，开发者也可以自定义折断点而支持更定制化的响应式设计用户体验。

注意 为实现不同分辨率下响应式设计的界面效果，这一章将会使用桌面浏览器Google Chrome来演示执行效果，而不是使用Android模拟器。在实际开发中，开发者也可以首先在桌面浏览器调试通过之后，再在移动设备浏览器中执行响应式设计的程序。

基于媒体查询技术的折断点技术实现响应式设计已经在桌面浏览器开发中被广泛使用了，大致的实现过程如下所示。

(1) 定义CSS的媒体查询和折断点。

(2) 通过流式分栏布局或者页面元素进行呈现。

(3) 在不同的媒体查询设置中，使用不同的图片和文字定义。

不管使用哪种方法实现响应式设计，开发者都需要投入更多精力在不同分辨率的设备中进行测试。例如，字体在高分辨率屏幕和普通移动设备屏幕中是否都显示正常，排版和布局在水平方向和垂直方向上是否都执行正常，等等。

通常，开发者在设计之初需要考虑到这样一些细节。

- 如果在页面中使用媒体查询技术，并根据移动设备分辨率的不同而设置不同的折断点，那么折断点的设置通常需要放在jQuery Mobile等CSS之后，否则将可能会不起作用。
- 需要根据相对长度单位em而不是根据像素设置截断点。一些移动设备分辨率非常高，而有的则没有那么高，如果按照像素设置折断点，则可能在高分辨率移动设备下，用户界面呈现异常，所以需要使用相对长度。
- 设置截断点的时候，不但需要考虑移动设备的分辨率，更需要考虑内容所占的尺寸。例如在分栏的时候，如果分栏的`div`设置不够高而内容很多，则在一些移动设备分辨率下，可能出现各个`div`的高度参差不齐。所以，如果内容文字较多，则需要根据内容篇幅考虑`div`所占的高度，以保证界面呈现效果。
- 从低分辨率的移动设备开始设计，通常更容易设计兼容性高的用户界面。如果低分辨率下能呈现正常，通常高分辨率下也没有问题。也正因为这样，在大多数媒体查询的CSS设计中，建议折断点要基于`min-width`参数进行设置。
- 对于支持多语言的场景，不同字体的尺寸是不同的。例如，在同样的内容下，中文所占的篇幅比英文小。如果使用相同的CSS设置和折断点设置，则可能英文下显示整齐，而中文环境下却显得松松垮垮，或者中文下显示整齐，而英文下却参差不齐。为此，有可能需要根据不同语种设计不同的CSS。这虽然并不是严格意义上的响应式设计的讨论范畴，但是却影响到响应式设计的折断点设置等细节的用户界面设计。

小经验　由于响应式设计需要能够支持多种不同的分辨率环境，如果直接在移动设备浏览器上进行调试，成本会很大。开发者可以首先在桌面浏览器（例如Firefox）中调试完成，然后在移动设备浏览器中进行集成和测试。

12.2　分栏技术

在jQuery Mobile 1.3.0出现之前，实现的分栏页面通常具有固定的栏数。在jQuery Mobile 1.3.0之后，可以根据移动设备屏幕分辨率的不同，通过设置媒体查询的折断点，实现响应式设计的分栏界面。

在图12-1中，我们分别呈现了视口尺寸不同的界面呈现效果。值得指出的是，这些界面的代码都是同一套的。



图12-1　响应式设计的分栏技术呈现效果

在图12-1中，当视口尺寸最小的时候，各个分栏的内容从上而下依次排开。当视口尺寸略大一些时，第一个分栏独立一行，而其余分栏排列在第二行。当视口尺寸再大时，则一行之中水平排列了4个分栏。随着页面的视口尺寸继续增大，布局没有发生变化，但是各个分栏中的文字尺寸比之前大了很多。这是一种典型的设计：随着视口尺寸的变化，界面也会发生相应的变化。在不同的视口尺寸下，用户界面呈现不同的分栏布局，字号也随之改变。

在实现这样的分栏设计时，需要根据不同的视口尺寸定义不同的折断点以及相应的各个分栏所占的宽度。当一行占满的时候，自然而然启用下一行，这就实现了不同的分栏效果。

以图12-1中第二个呈现界面为例，媒体查询的折断点设置为`min-width: 24em`，其中的CSS设置如下：

```
/*最小宽度为24em的时候，使用这部分CSS设定*/
 @media all and (min-width: 24em) {
  .responsive-grids div {
    min-height: 7em;
  }
  .responsive-grids .ui-block-b, .responsive-grids .ui-block-c, .responsive-grids .ui-block-d {
      float: left;
    width: 33.2%;
  }
  .responsive-grids .ui-block-b p, .responsive-grids .ui-block-c p, .responsive-grids .ui-block-c p {
    font-size: .5em;
  }
}
```

下面大概解释一下这段CSS的定义。

- 分栏的最小高度为 7em。
- 第一行分栏`ui-block-a`的宽度为默认宽度100%。
- 第二行分栏为`ui-block-b`、`ui-block-c`和`ui-block-d`，各个分栏的宽度为33.2%。正好这三个排列在一起，大致为屏幕的全部宽度。
- 文字尺寸略小。

其他折断点下的CSS设置与这个大致相当。只是在视口最小的时候，建议将所有分栏明确设置为宽度100%，这样能确保显示和布局正确。

完整的响应式设计的分栏技术呈现代码如代码清单12-1所示。

代码清单12-1　响应式设计的分栏技术

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <style>
```

```
.responsive-grids div {
  text-align: left;
  border-color: #ddd;
}
.responsive-grids p {
  color: #777;
  line-height: 140%
}
/* Stack all blocks to start */
.responsive-grids .ui-block-a, .responsive-grids .ui-block-b, .responsive-grids
  .ui-block-c .responsive-grids .ui-block-d {
  width: 100%;
  float: none;
}
.responsive-grids .ui-block-b, .responsive-grids .ui-block-c, .responsive-grids
  .ui-block-d {
  float: left;
  width: 100%;
}

/*最小宽度为24em的时候，使用这部分CSS设定*/
@media all and (min-width: 24em) {
  .responsive-grids div {
    min-height: 7em;
  }
  .responsive-grids .ui-block-b, .responsive-grids .ui-block-c, .responsive-grids
    .ui-block-d {
    float: left;
    width: 33.2%;
  }
  .responsive-grids .ui-block-b p, .responsive-grids .ui-block-c p,
    .responsive-grids .ui-block-c p {
    font-size: .5em;
  }
}

/*最小宽度为55em的时候，使用这部分CSS设定*/
@media all and (min-width: 55em) {
  .responsive-grids div {
    min-height: 7em;
  }
  .responsive-grids .ui-block-a, .responsive-grids .ui-block-c {
    float: left;
    width: 48.95%;
  }
  .responsive-grids .ui-block-b, .responsive-grids .ui-block-c, .responsive-grids
    .ui-block-d {
    float: left;
    width: 12.925%;
  }
}

/*最小宽度为68em的时候，使用这部分CSS设定，此时文字变大了*/
@media all and (min-width: 68em) {
```

```
        .responsive-grids {
          font-size: 125%;
        }
        .responsive-grids .ui-block-a, .responsive-grids .ui-block-c {
          float: left;
          width: 48.95%;
        }
        .responsive-grids .ui-block-b, .responsive-grids .ui-block-c, .responsive-grids
          .ui-block-d {
          float: left;
          width: 12.925%;
        }
      }
    </style>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="分栏布局的响应式设计">
      <header data-role="header">
        <h1>分栏布局的响应式设计</h1>
      </header>
      <div class="content" data-role="content"> 在响应式设计的分栏布局中，不同的折断点宽度下，
        所使用的栏目宽度、布局以及字体、字号、颜色都是可以定制的。
        <div style="height:15px;"></div>
        <div class="responsive-grids">
          <div class="ui-block-a">
            <div class="ui-body ui-body-c">
              <h1>第1栏</h1>
              第1栏的内容
            </div>
          </div>
          <div class="ui-block-b">
            <div class="ui-body ui-body-c">
              <h2>第2栏</h2>
              第2栏的内容
            </div>
          </div>
          <div class="ui-block-c">
            <div class="ui-body ui-body-c">
              <h3>第3栏</h3>
              第3栏的内容
            </div>
          </div>
          <div class="ui-block-d">
            <div class="ui-body ui-body-c">
              <h4>第4栏</h4>
              第4栏的内容
            </div>
          </div>
        </div>
      </div>
    </section>
  </body>
</html>
```

注意　响应式设计的CSS定义需要在页面各个CSS库（特别是jQuery Mobile库）之后定义，否则可能会不生效。开发者需要小心尝试不同的视口尺寸，以确定分栏的最小高度的设置是否正确。

12.3　回流表格

在使用HTML5之前，前端设计师往往会尽量避免页面出现回流（reflow）和重绘（repaint）的情况。因为回流和重绘很可能导致页面呈现速度变慢，并影响用户体验。而在基于HTML5的jQuery Mobile应用开发中，回流技术已成为一种响应式设计方法。

基于回流所绘制的表格，当视口尺寸比较宽的时候，表格所有的字段从左到右依次排列。而当视口尺寸比较小的时候，各个字段则变为从上到下依次排列。这种变化也是一种表格的响应式设计。

在图12-2中，左图是视口尺寸较宽时的界面呈现效果，右图是视口较窄时的呈现效果。



图12-2　不同视口宽度下的回流表格呈现效果

实现回流表格的方法非常简单：在表格声明中设置`data-role`为`table`，`data-mode`属性为`reflow`，并设置`class`为`ui-responsive`就可以了。

图12-2的实现代码可参见代码清单12-2。

代码清单12-2　回流表格

```
<table data-role="table" id="movie-table" data-mode="reflow" class="ui-responsive table-stroke">
  <thead>
    <tr>
      <th data-priority="1"><abbr title="营销系列编号">#</abbr></th>
      <th data-priority="persist">营销系列</th>
      <th data-priority="2">展示率</th>
      <th data-priority="3">点击</th>
      <th data-priority="4">费用</th>
    </tr>
  </thead>
  <tbody>
    <tr>
```

```
      <th>1</th>
      <td>广告系列A</td>
      <td>1,000</td>
      <td>10</td>
      <td>$10.00</td>
    </tr>
    <tr>
      <th>2</th>
      <td>广告系列B</td>
      <td>500</td>
      <td>9</td>
      <td>$6.00</td>
    </tr>
    <tr>
      <th>3</th>
      <td>广告系列D</td>
      <td>4,000</td>
      <td>90</td>
      <td>$89.00</td>
    </tr>
    <tr>
      <th>4</th>
      <td>广告系列F</td>
      <td>60,000</td>
      <td>1,500</td>
      <td>$980.00</td>
    </tr>
  </tbody>
</table>
```

12.4　字段切换表格

当在移动设备浏览器中呈现字段切换表格时，jQuery Mobile会检查视口的尺寸。如果尺寸较大，则可以呈现表格的所有字段。如果尺寸较小，则会呈现高优先级的字段，而自动隐藏低优先级的字段。能够自动根据视口尺寸而选择显示或隐藏表格字段，是字段切换表格的主要特点。图12-3是不同视口尺寸下字段切换表格的呈现效果，左图的视口宽度较宽，而右图较窄。



图12-3　字段切换表格

在使用字段切换表格时，表格右上角有一个用于选择字段的Columns菜单，如果有字段被隐藏起来，点击这个按钮即可在菜单中选择再次显示的字段，如图12-4所示。



图12-4 通过Columns菜单呈现隐藏的字段

注意 当视口尺寸从大到小变化时，表格中的字段会被自动隐藏掉。而当视口从小到大变化的过程中，被隐藏的字段可能不会自动显示出来，此时需要手工通过Columns菜单将字段呈现出来，或者重新刷新浏览器将其呈现出来。

实现字段切换表格的方法非常简单，只需要在表格的容器中声明data-role为table，data-mode为columntoggle，并设置class为ui-responsive就可以了，相关代码如下：

```
<table data-role="table" id="movie-table" data-mode="columntoggle" class="ui-responsive table-stroke">
  ...
</table>
```

12.5 滑动面板

通常，jQuery Mobile界面是从上到下在移动设备浏览器中依次排列展开的。滑动面板则不同，它在移动设备浏览器的左侧或者右侧展开。使用者可以在滑动面板中进行操作，操作完成后再将滑动面板折叠回去。

滑动面板的用法非常类似于对话框，我们可以将表单、列表、菜单或者介绍文字集成在滑动面板中。图12-5是滑动面板展开和关闭时的呈现效果。点击左图中的超级链接，滑动面板从屏幕左侧弹出。输入完消息并点击“发送”按钮后，滑动面板再次关闭起来。



图12-5 滑动面板呈现效果

实现滑动面板时，需要在页面中加入滑动面板的容器，如下所示：

```
<div data-role="panel" id="sliding-panel">
  <!-- 此处为滑动面板的内容 -->
</div>
```

滑动面板是一个独立在页面、页脚和正文之外的独立容器。值得注意的是，滑动面板的容器可以写在页面开始或者结束的位置，但不要写在页眉、正文或者页脚之中。

如果要打开一个滑动面板，可以通过超级链接或者超级链接按钮来实现，例如：

```
<a href="#message-panel">
  打开左侧面板，发送消息。
</a>
```

如果要在程序中关闭滑动面板，则可以在超级链接中声明`data-rel`为`close`或者通过JavaScript的`Close`方法来关闭。例如，在滑动面板内部关闭滑动面板的代码是这样的：

```
<a href="#" data-rel="close" data-role="button" data-theme="c" data-mini="true">
  取消
</a>
```

图12-5所对应的完整的滑动面板程序如代码清单12-3所示。

代码清单12-3 滑动面板

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
```

```
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="面板">
      <header data-role="header">
        <h1>面板</h1>
      </header>
      <div class="content" data-role="content"> 点击打开面板按钮，可以打开面板。
        <div style="height:15px;"></div>
        <a href="#message-panel">打开左侧面板，发送消息。</a> </div>
      <div data-role="panel" data-position-fixed="true" data-theme="a" id="message-panel">
        <form class="userform">
          <h2>发消息</h2>
          <label for="name">Lu Ming的消息:</label>
          <textarea cols="20" rows="10" name="message" id="message"></textarea>
          <div class="ui-grid-a">
            <div class="ui-block-a">
              <a href="#" data-rel="close" data-role="button" data-theme="c" data- mini="true">
                取消
              </a>
            </div>
            <div class="ui-block-b">
              <a href="#" data-rel="close" data-role="button" data-theme="b" data-mini="true">
                发送
              </a>
            </div>
          </div>
        </form>
      </div>
    </section>
  </body>
</html>
```

12.6　支持触控操作的滑动面板

通常，移动设备浏览器的尺寸有限，而有的内容却很长，这时如果需要在页面中找到打开或者关闭滑动面板的按钮，用户将可能需要翻屏好几次才能定位到。如果可以使用横向的轻扫屏幕将滑动面板调出来，将会方便使用者操作。

在jQuery Mobile中，横向轻扫可以使用swiperight或者swipeleft事件实现。当事件触发的时候，调用相应面板的open函数，就可以打开这个滑动面板了。这个实现方法的JavaScript代码片段如代码清单12-4所示。

代码清单12-4　滑动面板

```
<script type='text/javascript'>
  $('#MainPage').live('swiperight', function(){
    $('#message-panel').panel('open');
  });
</script>
```

第 13 章

主题风格美化

13

在jQuery Mobile中，我们可以对页面样式和页面中的工具栏、按钮、表单元素、列表进行颜色设定。jQuery Mobile默认内置了5种不同的配色色版，开发者也可以通过ThemeRoller定义适合自己应用程序的页面主题风格。

在这一章中，我们将接触到：

- 主题与色版；
- 内置色版；
- 通过ThemeRoller自定义主题；
- 高级开发技术。

13.1 主题与色版

在jQuery Mobile中，我们可以通过定义主题风格来定义界面样式。主题通常包括界面元素样式以及色版配色方案。

基于jQuery Mobile开发移动应用时，我们可以设置特定的界面元素样式，例如某个界面元素的尺寸，将其边框设置为直角矩形边框或者圆角矩形边框等。而色版用于配置界面元素的配色。

对jQuery Mobile应用进行个性化界面设计时，大多情况下就是进行主题和色版的个性化设置，除非涉及响应式设计或一些特殊的用户界面个性化场景。在色版之外，对于主题的定义通常是字体，激活状态的文字、边界和背景设置，圆角矩形按钮或分组的边框样式，图标样式和阴影效果等。

通常，jQuery Mobile的内置主题样式和色版一起保存在层叠样式表中。下面的代码将jQuery Mobile默认的层叠样式表引入页面中：

```
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.1/jquery.mobile-1.3.1.min.css" />
```

小经验 为了提升用户访问速度，我们建议将CSS文件、JavaScript文件和资源图片存储在国内服务器上。此外，最好同时采购和配置CDN服务，这样打开应用的速度将会更快。

13.2　内置色版

在jQuery Mobile中，默认的层叠样式表内置了5种不同的色版。对于不同的页面元素，我们可以通过设置不同的色版来改变样式。

图13-1分别为色版a和色版c的呈现效果，两张截屏同为登录界面，左图采用色版a作为页面的色版，而右图采用色版c。色版设置不同，界面元素的外形、尺寸等不变，但是颜色发生了很大变化。



图13-1　左图使用色版a，右图使用色版c

这是左图色版设置的代码片段：

```
<section id="MainPage" data-role="page" data-title="登录表单" data-theme="a">
  <!—这里加入登录表单 -->
</section>
```

右图和左图代码中的唯一不同在于data-theme属性的设定，右图中将该属性设定为色版c，相关代码如下：

```
<section id="MainPage" data-role="page" data-title="登录表单" data-theme="c">
  <!—这里加入登录表单 -->
</section>
```

由于page容器的色版发生变化，其中的工具栏、输入框、按钮的配色都随之改变。可见，父级容器中色版的变化将对界面主题样式产生直接影响。

在jQuery Mobile中，默认色版定义可以自a到e，共有5种不同的样式。开发者可以自行书写代码以测试，在此不再赘述。

通常，我们可以自定义页面元素所使用的色版。如果没有自定义的色版，则会取自上一级容器的色版。在第10章的登录界面中，“提交”与“取消”按钮使用了不同的颜色便是基于此，如图13-2所示。



图13-2 通过data-theme属性设定按钮主题

设置特定的data-theme样式之后，其主题配色将不再继承自父级容器，而是用自定义的配色样式。在代码清单13-1中，“取消”按钮使用了data-theme为d的主题风格，按钮颜色为白底黑字，“提交”按钮使用了data-theme为a的主题风格，按钮颜色为黑底白字。

代码清单13-1 设置表单元素的主题

```
<fieldset class="ui-grid-a">
  <div class="ui-block-a">
    <button type="reset" data-theme="d">取消</button>
  </div>
  <div class="ui-block-b">
    <button type="submit" data-theme="a">提交</button>
  </div>
</fieldset>
```

除了按钮之外，其他的表单元素（比如工具栏和列表）等均可以通过设置主题样式而设定不同的颜色。

13.3 通过 ThemeRoller 自定义主题

很多场合下，开发者希望根据自己产品的需要为移动应用定制个性化的配色方案。如果开发者自己开发这样的主题和色版，将会是一件非常耗时的工作，而jQuery Mobile提供了ThemeRoller工具，这个工具能够帮助开发者快速定义26种色版，以适应不同的应用场景。

13.3.1 ThemeRoller的基本概念

ThemeRoller（如图13-3所示）是一个在线服务，开发者可以在线进行jQuery Mobile色版定制、阅览、下载与分享，也可以将开发完成的色版文件导入到ThemeRoller以进行再编辑。

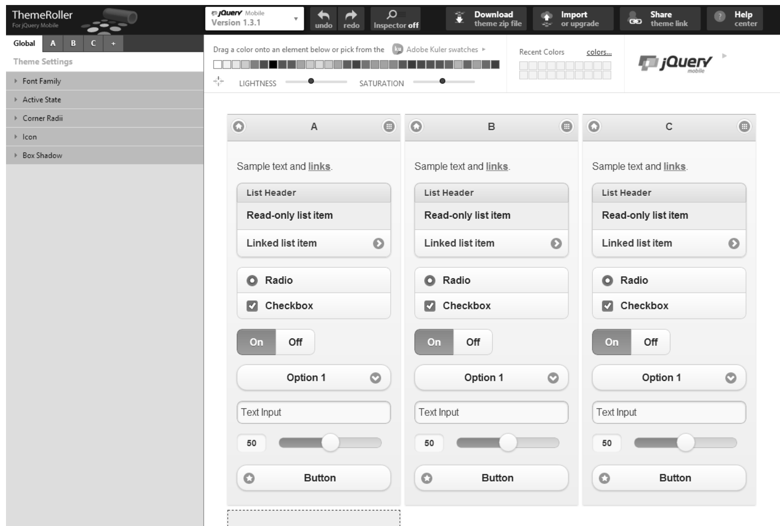


图13-3　ThemeRoller

在使用ThemeRoller进行界面定义的时候，首先需要注意所使用的jQuery Mobile版本号。在ThemeRoller上方，有选择jQuery Mobile版本的下拉列表，如图13-4所示。这里需要说明的是，ThemeRoller中jQuery Mobile的版本号需要和生产系统中的jQuery Mobile版本号一致。如果开发者使用jQuery Mobile 1.1.0版本，则在ThemeRoller中选择Version 1.1.0。

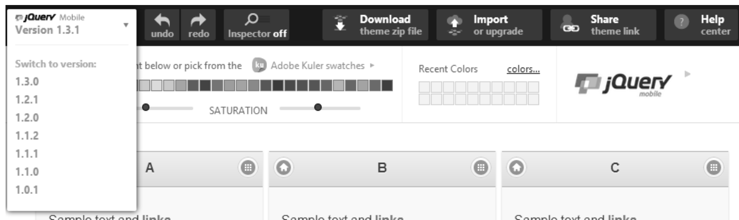


图13-4　jQuery Mobile版本下拉列表

在ThemeRoller界面中，主要由4部分组成，具体如表13-1所示。

表13-1　ThemeRoller的界面布局

布　　局	功　　能
功能按钮	位于ThemeRoller最上方，用于实现版本号选择、恢复与重做，打开与关闭Inspector，下载主题文件，导入主题文件，分享自定义主题链接与帮助功能
检查窗格（Inspector）	位于图13-3屏幕左侧，用于实现全局设置和特定色版设置
QuickSwatch栏	位于图13-3中屏幕右侧功能按钮和预览窗格之间，用于将颜色拖曳到特定页面元素，实现快速设置色版的功能
预览窗格	图13-3中屏幕右侧大片区域，默认包含了3个移动应用界面，预览所设定主题风格的呈现样式

13.3.2　编辑全局设置与色版

在检查窗格部分，可以进行主题的全局设置和特定色版的设置，如图13-5所示。

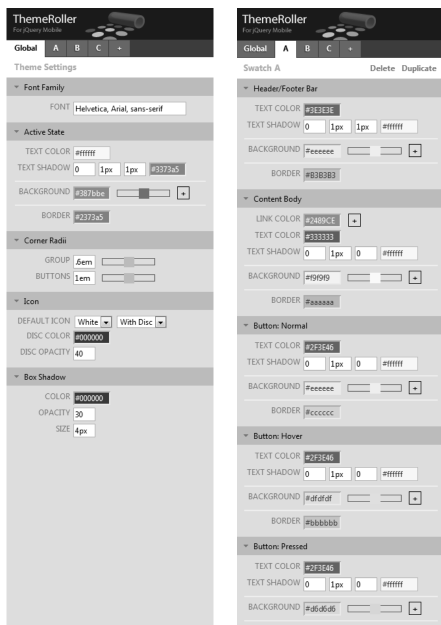


图13-5 左图为全局设置，右图为色版a的设置

通过全局设置，我们可以设置：

- ❑ 字体；
- ❑ 激活状态，例如文字颜色、阴影效果、背景颜色和边框颜色等；
- ❑ 圆角设置，例如分组和按钮的圆角尺寸；
- ❑ 图标样式；
- ❑ 阴影。

在jQuery Mobile中，默认包含了5种不同的色版，分别以a至e来表示。通过ThemeRoller，我们可以自定义26种色版，分别通过a至z来表示。

点击检查窗格右侧的+按钮，或者在预览窗格中点击Add swatch…按钮，则可以添加新的色版。检查窗格上的色版A、B、C、D……分别对应到程序中所使用的色版定义。例如，在页面中加载自定义色版F中的样式的代码如代码清单13-2所示。

代码清单13-2　设定自定义风格

```
<section id="MainPage" data-role="page" data-title="登录表单" data-theme="f">
  <!-- 这里是页面内容 -->
</section>
```

当一个色版定义完成后，我们希望复制当前色版并在当前色版的基础上开发新的色版，此时直接点击色版上方的Duplicate链接即可。

如果需要删除当前色版，则点击Delete链接即可。

基于ThemeRoller进行色版设计，可以定义这样一些内容：

- 页眉和页脚；
- 页面内容；
- 按钮样式。

jQuery Mobile初学者在色版开发中往往容易混淆全局设置和特定色版设置，这是因为它们都有关于一些特定界面颜色的设置。如果设置错误，可以使用功能按钮中的undo恢复之前的操作。

13.3.3　导入、下载和分享自定义色版

对编辑完的色版进行再次编辑，可以将主题CSS文件导入到ThemeRoller中进行。编辑完成后，我们可以直接将其下载到本地。

点击Import or upgrade按钮，在打开的Import Theme对话框中将之前定义好的色版CSS内容复制到对话框中，然后点击Import按钮以实现导入操作，如图13-6所示。

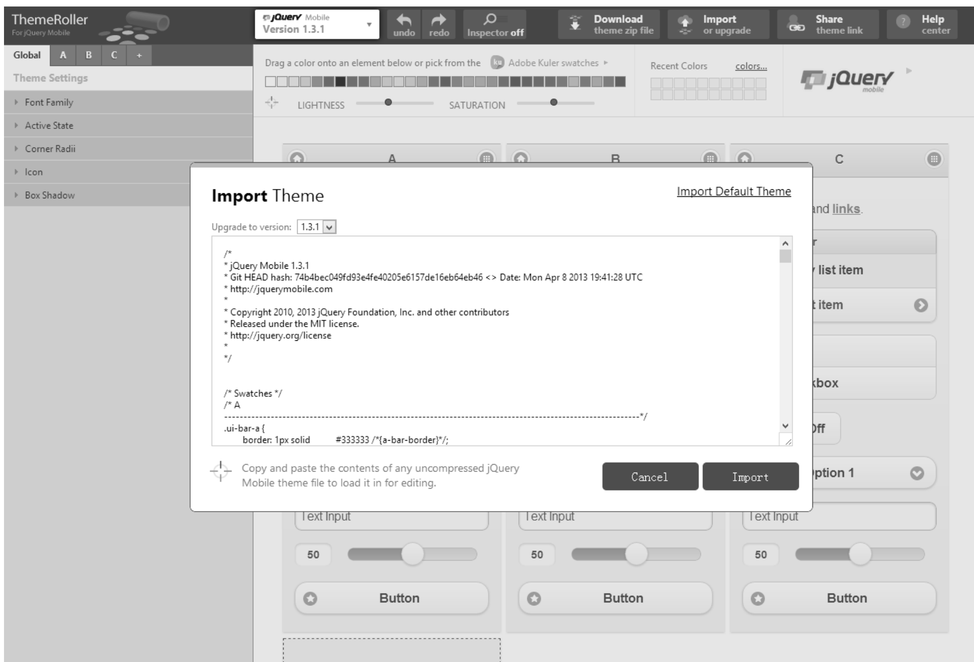


图13-6　Import Theme对话框

图13-7是将jQuery Mobile默认的主题定义文件导入到ThemeRoller后的界面。

图13-7　导入默认主题样式的效果

完成色版设置后，可以在ThemeRoller的下载界面中，输入Theme Name后，将其打包为ZIP格式下载，如图13-8所示。

图13-8　下载主题

另外，在下载主题的界面中，ThemeRoller使用黄色粗体高亮显示出集成定制主题层叠样式表的方法。需要注意的是，在使用自定义主题时，一定要引用jquery.mobile.structure这个层叠样式表定义，示例代码如代码清单13-3所示。jquery.mobile.structure这个层叠样式表文件是jQuery Mobile库的一部分，可以在下载的jQuery Mobile压缩文件中找到。

代码清单13-3　加载自定义主题

```
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>跨平台移动应用</title>
  <link rel="stylesheet" href="../css/themes/my-custom-theme.css" />
  <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile.structure-1.3.1.min.css" />
  <script src="../js/jquery-1.7.1.min.js"></script>
  <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
</head>
```

主题的设计也可以通过超级链接进行分享。点击Share按钮，ThemeRoller将会保存当前主题的设定，并呈现分享链接，而开发者可以将生成的超级链接发送给其他人，如图13-9所示。接收到这个分享链接并打开时，ThemeRoller将自动呈现前次分享的内容，而接收者可以基于这样的分享继续进行编辑和设计。

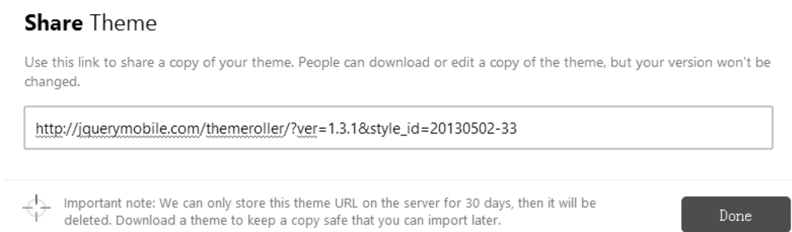


图13-9　分享主题

13.4　高级开发技术

在进行主题开发的过程中，我们不但可以在页面加载时设定某个页面、按钮或表单元素的主题样式，也可以通过程序设定它们。

除了可以通过data-theme属性设定主题之外，jQuery Mobile还提供了一套通过CSS样式设定主题风格的方法。开发者可以通过声明CSS样式的方式设定页面和页面元素的主题。主要的CSS样式定义如表13-2所示。

表13-2　页面主题的CSS样式定义

CSS属性	功　　能
ui-bar-(a-z)	用于设定工具栏的主题风格，例如ui-bar-a
ui-body-(a-z)	用于设定页面和页面元素的主题风格，例如ui-body-a
ui-btn-up-(a-z)	用于设定按钮的主题风格，例如ui-btn-up-a
ui-corner-all	用于设定圆角矩形边框
ui-shadow	用于设定阴影效果
ui-disabled	用于设定为禁用效果

在程序执行期间，动态调整页面和页面元素主题设置的步骤如下。

(1) 获取需要调整主题设定的DOM对象以及这个DOM对象当前的主题设定。

(2) 重新设置DOM对象的主题，并移除掉CSS样式中当前设置的色版。

(3) 添加新的色版对应的CSS样式到DOM对象中。

(4) 通过create事件使得主题设定与CSS样式设定生效。

实现动态调整页面元素主题的代码片段如代码清单13-4所示。

代码清单13-4　动态调整页面元素主题

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>跨平台移动应用</title>
    <link rel="stylesheet" href="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.css" />
    <script src="../js/jquery-1.7.1.min.js"></script>
    <script src="../js/jquery.mobile-1.3.1/jquery.mobile-1.3.1.min.js"></script>
    <script type="text/javascript">
      function ChangeSwatche(targetSwatche)
      {
        var targetDOM = $('#swatchDemo');
        var currentSwatche  = targetDOM.attr('data-theme');

        targetDOM.attr('data-theme',targetSwatche).removeClass('ui-body-'+currentSwatche);
        targetDOM.addClass('ui-body-'+targetSwatche);
        targetDOM.trigger('create');
      }
    </script>
  </head>
  <body>
    <section id="MainPage" data-role="page" data-title="更换色版">
      <header data-role="header">
        <h1>更换色版</h1>
      </header>
      <div class="content" data-role="content">
        <form>
        <div data-role="fieldcontain">
          <input type="text" name="swatchDemo" id="swatchDemo"
            value="" placeholder="更换色版" data-theme="e" />
          <div style="margin-top:20px;" >
          <fieldset class="ui-grid-a">
              <div class="ui-block-a">
                <button type="button" onClick="return ChangeSwatche('a');">
                  色版A
                </button>
              </div>
              <div class="ui-block-b">
                <button type="button" onClick="return ChangeSwatche('c');">
                  色版C
                </button>
```

```
                </div>
              </fieldset>
            </div>
          </form>
        </div>
      </section>
    </body>
  </html>
```

在这段代码中，我们将需要设置的色版参数传入ChangeSwatche函数中。ChangeSwatche函数将id为swatchDemo的文本框的原有主题设定移除掉，再设定新的主题。然后，通过触发create事件使得新设定的主题生效。实现效果如图13-10所示。



图13-10　左图为“色版A”按钮按下时的效果，右图为“色版C”按钮按下时的效果

注意　当基于属性或选项设定页面主题时，如果页面或表单元素不声明特定的主题，则会继承父级容器的主题。例如，页面主题为e，如果不进行特别设定，则页面中所有元素的主题均为e。 但是通过JavaScript进行设定时，当父级容器的主题被重新设定时，其内部所包含的页面或表单元素并不会一起更改，除非遍历所有页面或表单元素逐一重新设定主题。

附录 A



JavaScript测试自动化

基于测试驱动开发和测试自动化技术开展测试活动，将有助于快速稳定产品质量，改善产品发布效率，从而缩短产品开发周期，降低开发成本。使用jQuery Mobile开发的移动应用基于移动设备浏览器和桌面浏览器环境来运行,而大多数的JavaScript测试自动化工具基于桌面浏览器来运行，所以，开发者可以将jQuery Mobile应用首先运行在桌面浏览器中，通过自动化技术保证程序运行良好，然后再在移动设备中进一步测试，这样整体的测试成本将会更低，效率更高。

在这一章中，我们将接触到：

- 基于QUnit开发JavaScript测试驱动；
- QUnit断言；
- 其他测试技术。

A.1 基于 QUnit 开发 JavaScript 测试驱动

测试驱动是通过测试来推动软件开发活动开展的一种实践方法,能够帮助开发者更加聚焦于所要实现的功能而减少项目“镀金”，可以帮助开发者从多种角度对所需要开发的模块开展思辨，完善设计，以达到完善开发交付物的目的。在以极限编程为代表的敏捷软件开发中，测试驱动是一种旨在提升软件质量、改善质量成本结构的最佳实践。

通常，在测试驱动开发中，工程师在开发一个应用模块之前，先基于设计开发测试程序，再通过编码使得开发的模块通过测试驱动测试。

QUnit是一种面向JavaScript的测试驱动框架，使用它，开发人员可以对各种应用场景开展测试，甚至QUnit所提供的异步方法还能帮助开发人员测试Ajax环境下JavaScript程序的运行状况。在jQuery Mobile的实际应用中，为了改善用户体验，大量数据交互往往通过Ajax实现。

除了JavaScript的单元测试工具QUnit外，其他类似的单元测试工具还有很多，例如专门面向C/C++的测试驱动CppUnit、专门面向Java的测试驱动JUnit以及专门面向.NET的测试驱动NUnit和MSUnit，等等。

使用QUnit进行JavaScript单元测试的效果如图A-1所示。

在Hello World的这个测试驱动设计中，包含了两个基本的测试，前一个执行成功，后一个执行失败。默认情况下，执行成功的用例会折叠起来，而执行失败的测试用例会被展开。



图A-1　左图为在IE中运行QUnit的效果，右图为在Android中运行QUnit的效果

这个Hello World测试用例的完整代码如代码清单A-1所示。

代码清单A-1　QUnit的Hello World测试用例

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <link rel="stylesheet" href="http://code.jquery.com/qunit/git/qunit.css"
      type="text/css" media="screen" />
    <script type="text/javascript" src="http://code.jquery.com/qunit/git/qunit.js"></script>
    <script>
      $(document).ready(function(){
        module("测试成功的用例");
        test("QUnit 测试的 Hello World", function() {
          var actual = "Hello";
          equal( actual, "Hello", "期待结果是Hello" );
        });

        module("测试失败的用例");
        test("QUnit 测试的 Hello World", function() {
          var value = "Hello";
          equal( value, "World", "期待结果是World，这里是Hello。QUnit会报告测试失败。" );
        });
      });
    </script>
  </head>
  <body>
    <h1 id="qunit-header">QUnit JavaScript单元测试</h1>
    <h2 id="qunit-banner"></h2>
```

```
    <div id="qunit-testrunner-toolbar"></div>
    <h2 id="qunit-userAgent"></h2>
    <ol id="qunit-tests">
    </ol>
    <div id="qunit-fixture">测试标记</div>
  </body>
</html>
```

开发基于QUnit的测试驱动时，首先需要引用相关的JavaScript库。由于QUnit是基于jQuery所开发的，所以需要引用jQuery库、QUnit库和CSS文件，示例代码如下：

```
<script src="http://code.jquery.com/jquery-latest.js"></script>
<link rel="stylesheet" href="http://code.jquery.com/qunit/git/qunit.css" type="text/css" media="screen" />
<script type="text/javascript" src="http://code.jquery.com/qunit/git/qunit.js"></script>
```

值得注意的是，如果测试用例需要运行在移动设备中（例如Android或者iPhone），最好能够加入viewport的meta标签，否则可能会显示异常：

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

这个meta信息将会控制呈现内容的大小，保证内容中的文字按照正常比例显示，这也与惯常的jQuery Mobile程序一致。

在QUnit的HTML中定义有不同的DOM id，它们的用途各不相同：

```
<h1 id="qunit-header">QUnit JavaScript单元测试</h1>
<h2 id="qunit-banner"></h2>
<div id="qunit-testrunner-toolbar"></div>
<h2 id="qunit-userAgent"></h2>
<ol id="qunit-tests">
</ol>
<div id="qunit-fixture">测试标记</div>
```

其基本含义如表A-1所示。

表A-1　QUnit DOM容器id及功能

DOM容器id	功　　能
qunit-header	QUnit名称
qunit-banner	QUnit测试界面中位于测试标题下面的一条比较窄的横幅。如果QUnit执行成功，这个横幅是绿色的，如果失败则为红色。如果QUnit中包含大量单元测试，则可以通过这个横幅的颜色快速了解测试驱动执行中是否存在失败
quint-testrunner-toolbar	测试驱动执行环境的工具栏。例如，当点击Hide passed tests复选框时，只有失败的单元测试结果被呈现出来，而执行成功的单元测试被隐藏起来
qunit-userAgent	用以呈现测试环境。由于不同浏览器下JavaScript可能存在兼容性问题，这个属性将有助于辨识不同测试环境对于测试驱动执行状况的影响
qunit-tests	这是一个无符号列表的DOM容器，测试结果将会被输出在这里
quit-fixture	测试标记。当执行reset()函数时，将会被重置

在前面的Hello World!中，各个测试用例被包装在jQuery的ready()事件中。当页面和JavaScript加载完成时，ready()事件被触发，包装在ready()事件中的测试程序被依次执行。

在下面这段JavaScript代码中，module()函数标记了单元测试的模块，其中每个模块会有一个特定的名称：

```
module("测试成功的用例");
test("QUnit 测试的 Hello World", function() {
  var actual = "Hello";
  equal( actual, "Hello", "期待结果是Hello" );
});
```

这个名称将会出现在测试结果中。test()函数定义了测试用例的名称，预期执行断言的数量与测试用例。预期执行断言的数量通常可以省略掉。equal()是QUnit执行的断言。通过断言，QUnit比较执行结果与预期是否一致，如果一致则返回执行通过的消息。如果执行失败，则会输出预期值，实际值以及之间的差异。一旦断言错误，那么QUnit会使用红色标记这个错误的执行结果。

在每个执行结果的的后面，会有一个括号包含有3个数字，如图A-2所示。



图A-2　QUnit执行结果

在这3个数字中，第一个数字为执行失败的断言数量，第二个是执行成功的断言数量，第三个是在这个测试用例中所有被执行的断言数量。

如果test()函数中声明了预期的执行断言数量，那么这个计数也将作为一个断言在所有单元测试执行完成之后来进行比较。如果预期执行数量和实际执行的断言数量不一致，那么测试用例也会报错，这是一个比较有效的验证测试路径和测试覆盖情况的方法。特别是在执行验收测试自动化或者回归测试自动化的时候，这能有效提升测试质量。

A.2　QUnit 断言

使用QUnit从事开发时，经常会用到的断言如表A-2所示。

表A-2　QUnit断言及功能

断　　言	功　　能
ok(state, message)	验证state的布尔类型数值，如果state为true则测试通过，为false则测试失败
equal(actual, expected, message)	比较actual和expected数值是否一致，如果相同则测试通过，不同则测试失败
notEqual(actual, expected, message)	比较actual和expected数值是否不一致，如果不一致则测试通过，一致则测试失败
deepEqual(actual, expected, message)	比较对象或数组各个内容是否一致，如果一致则测试通过，不一致则测试失败
notDeepEqual(actual, expected, message)	比较对象或数组各个内容是否不一致，如果不一致则测试通过，一致则测试失败
strictEqual(actual, expected, message)	比较actual和excepted数值和类型是否一致，如果一致则测试通过，不一致则测试失败

（续）

断　　言	功　　能
notStrictEqual(actual, expected, message)	比较actual和excepted数值和类型是否不一致，如果不一致则测试通过，一致则测试失败
raises(block, expected, message)	验证代码段执行是否会触发异常，如果触发异常，则测试通过，否则测试失败

A.3 其他测试技术

如果测试环境比较复杂，除QUnit之外，还可能用到其他的一些JavaScript测试自动化工具。

Selenium是一种基于浏览器的测试自动化工具。基于Selenium可以方便地将测试分发到不同的测试环境，并根据测试脚本开展测试自动化活动。这样的过程能有效加强回归测试与验收测试自动化，有助于改善测试效率。jQuery Mobile应用可能在多种不同的移动设备和浏览器中执行，通过部署和使用Selenium将可以有效提升不同测试环境下的测试自动化，改善质量成本结构，优化移动应用开发成本结构，改善产品交付周期。

基于Selenium可以对如下环境开展测试自动化：

- Google Chrome 12.0+；
- Internet Explorer 6.0+；
- Firefox 3.0+；
- Android 2.3+；
- iOS 3+的iPhone环境以及iOS 3.2+的iPad环境。

开发Selenium测试自动化用例的开发语言可以是Java、C#、Python、Ruby、PHP和Perl等。这对于很多工程师而言，可以很方便地上手并开展测试活动。

Selenium项目位于http://seleniumhq.org/，开发者可以从这里下载最新的Selenium安装程序。

Sinon.JS是一种JavaScript单元测试工具集，可以与QUnit集成之后进行单元测试。

Sinon.JS支持的桌面浏览器包括：

- Internet Explorer 6.0+；
- Firefox 2.0+；
- Google Chrome 7.0+；
- Opera 10.10+；
- Apple Safari 3.2+。

支持的移动操作系统包括：

- Apple iOS 3.2；
- Android 2.1+。

此外，开发者还可以基于Sinon.JS对Node.JS 0.2.x+所开发的服务器端应用开展测试活动。

Sinon.JS的作者是*Test-Driven JavaScript Development*的作者，拥有多年测试自动化经验。Sinon.JS项目位于http://sinonjs.org/，开发者可以从这里下载最新的测试工具。

jQuery Mobile
开发指南

JavaScript是互联网前端的重要语言，其重要性也延伸到了后端、游戏等。jQuery Mobile是一个简单实用的JavaScript框架，通过它，你可以迅速在不同的手机系统、浏览器实现你的网页和Web App。jQuery Mobile身为一个移动端框架，可以使网站上的表单和UI更好地支持触屏，也能通过Ajax和HTML5 pushState让你的网站更自然地更新内容。本书是一本迅速开发移动端网页的指南。

——谢子斌（@zibin），W3C HTML5中文兴趣小组主席

未来到底是Web App的天下，还是Native App的天下？就此乔布斯曾经说过，虽然现阶段Native App给了用户更好的体验，但是如果现在的开发者不能有效地利用Web技术，那他就落伍了。Native App导致的信息孤岛和适配困难问题，并不符合互联网的核心价值，解决这些问题的钥匙就在HTML5规范下的jQuery Mobile技术。而本书是建立在大量实践基础上的经验总结，作者充分了解初学者如何一步步成为jQuery Mobile专家的过程。这本书的重要价值在于通过知识点和案例的有效结合帮助读者重复这一过程。

——刘锋，《互联网进化论》作者

使用HTML5可以快速开发具有良好设备兼容性、质量稳定的Web移动应用，这已是不争的事实。jQuery Mobile是一种基于HTML5的Web移动应用用户界面系统，使用这种技术将能够进一步快速开发统一用户界面的HTML5移动应用。本书对于常用的开发场景娓娓道来，适合作为常备参考资料。书中的高级开发技巧具有很强的实用性，能帮助开发者快速解决很多现实开发场景中的常见问题。

——田爱娜，HTML5梦工场发起人

图灵社区：iTuring.cn
热线：(010)51095186转600

分类建议 计算机/移动开发/jQuery Mobile

人民邮电出版社网址：www.ptpress.com.cn



ISBN 978-7-115-34371-0
定价：49.00元

看完了

如果您对本书内容有疑问，可发邮件至contact@turingbook.com，会有编辑或作译者协助答疑。也可访问图灵社区，参与本书讨论。

如果是有关电子书的建议或问题，请联系专用客服邮箱：ebook@turingbook.com。

在这里可以找到我们：

微博 @图灵教育 : 好书、活动每日播报
微博 @图灵社区 : 电子书和好文章的消息
微博 @图灵新知 : 图灵教育的科普小组
微信 图灵访谈 : ituring_interview，讲述码农精彩人生
微信 图灵教育 : turingbooks